DEFENSE INFORMATION SYSTEMS AGENCY
P. O. BOX 549
FORT MEADE, MARYLAND 20755-0549

IN REPLY REFER TO: Joint Interoperability Test Command (JTE) 18 Jun 14

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Joint Interoperability Certification of the Cisco Unified Presence Server (CUPS) 8.6.5 with Cisco Jabber 9.2.6

References: (a) Department of Defense Instruction 8100.04, "DoD Unified Capabilities (UC)," 9 December 2010
(b) DoD CIO, Memorandum, "Interim Guidance for Interoperability of Information Technology (IT) and National Security Systems (NSS)," 27 March 2012
(c) through (f), see Enclosure 1

1. **Certification Authority.** References (a) and (b) establish the Joint Interoperability Test Command (JITC) as the Joint Interoperability Certification Authority for the UC products.

2. **Conditions of Certification.** The CUPS 8.6.5 with Cisco Jabber 9.2.6; hereinafter referred to as the System Under Test (SUT), meets the critical requirements of the Unified Capabilities Extensible Messaging and Presence Protocol (XMPP) Requirements and the Unified Capabilities Requirements (UCR), References (c) and (d), and is certified for joint use as a XMPP Client/Server with the conditions described in Table 1. This certification expires upon changes that affect interoperability, but no later than three years from the date of the UC Approved Products List (APL) memorandum.

Table 1. Conditions

Condition	Operational Impact	Remarks
UCR Waivers		
The SUT does not support IPv6. The Office of the Secretary of Defense (OSD) granted a waiver for IPv6 on 16 May 2013.	Minor	
Conditions of Fielding		
None.		
Open Test Discrepancies		
The SUT does not correctly respond to stream errors. Instead of responding with a stream error immediately and closing the stream, the SUT terminates the connection non-gracefully.	Minor	See note 1.
The SUT does not generate a new Client-to-Server stream. The SUT reuses the old stream instead.	Minor	See note 1.
The SUT does not include empty <required/> element in its advertisement of the SASL.	Minor	See note 1.
The SUT does not fully comply with SASL failure requirements. The SUT meets SASL error conditions outlined in RFC 3920 and not RFC 6120. The SUT does not allow a configurable number of retries.	Minor	See note 1.

Table 1. Conditions (continued)

Condition	Operational Impact	Remarks
The SUT does not fully meet the Deleting a Roster Item requirement. Presence stanza of type "unsubscribe" is not sent to contact. Instead, the stanza is silently dropped.	Minor	See note 1.
The SUT partially complies with Rules for Server Processing of Outbound Subscription Requests.	Minor	See note 1.
The SUT partially complies to Rules for Server Processing of Outbound Subscription Cancellation. Upon receiving the outbound subscription cancellation, the contact's server does not send a presence stanza of type "unavailable" from all of the contacts online resources to the user.	Minor	See note 2.
The SUT partially complies with Rules for Server Processing of Inbound Unsubscribe. The SUT doesn't check if the user is in the contact's roster with subscription='from' or subscription='both'.	Minor	See note 2.
The SUT does not comply with Server Generation of Outbound Presence Probe.	Minor	See note 1.
The SUT establishes SASL external authentication with incorrect domain.	Minor	See note 2.
The SUT does not comply with the requirements in XMPP Extension XEP-0045: Multi-User Chat.	Minor	See note 1.

NOTES:
1. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
2. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.

LEGEND:

DISA	Defense Information Systems Agency	SASL	Simple Authentication and Security Layer
IPv6	Internet Protocol version 6	SUT	System Under Test
RFC	Request for Comments	UCR	Unified Capabilities Requirements
POA&M	Plan of Action & Milestones	XMPP	Extensible Messaging and Presence Protocol

3. **Interoperability Status.** Table 2 provides the SUT interface interoperability status and Table 3 provides the Capability Requirements (CR) and Functional Requirements (FR) status. Table 4 provides the UC APL product summary.

Table 2. SUT Interface Status

Interface (See note 1.)	Threshold CR/FR Requirements (See note 2.)	Status	Remarks
Network Management Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1	Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1	Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1	Met	See note 3.
Server Network Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1, 2, 3	Partially Met	See note 3.
Client Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1, 2, 3	Partially Met	See note 3.

NOTES:
1. References (c) and (d) do not specify a minimum required Ethernet interface, therefore, any one of the listed interfaces can be supported.
2. The SUT high-level CR and FR ID numbers depicted in the Threshold CRs/FRs column can be cross-referenced in Table 3. These high-level CR/FR requirements refer to a detailed list of requirements provided in Enclosure 3.
3. The SUT does not support IPv6. The Office of the Secretary of Defense (OSD) granted a waiver for IPv6 on 16 May 2013.

Table 2. SUT Interface Status (continued)

LEGEND:			
802.3ab	1000BaseT Gbps Ethernet over twisted pair at 1 Gbps (125 Mbps)	FR	Functional Requirement
		ID	Identification
802.3i	10BaseT Mbps over twisted pair	IEEE	Institute of Electrical and Electronics Engineers
802.3u	Standard for 100 Mbps Ethernet	IPv6	Internet Protocol version 6
C	Conditional	SUT	System Under Test
CR	Capability Requirement	UTP	Unshielded Twisted Pair

Table 3. SUT Capability Requirements and Functional Requirements Status

CR/FR ID	XMPP and UCR Requirements (High-Level) (See note 1.)	UC XMPP Reference (See note 2.)	Status
1	XML Streams	2.6	Partially Met (See note 3.)
2	TLS and STARTTLS Negotiation	2.7	Met (See note 4.)
3	Authentication and SASL Negotiation	2.8	Partially Met (See note 3.)
4	Resource Binding	2.9	Met
5	XML Stanzas	2.10	Met
6	Roster Management	2.11	Met
7	Presence Subscription Management	2.12	Partially Met
8	Exchanging Presence Information	2.13	Partially Met
9	Exchanging Messaging	2.14	Met
10	Conformance Requirements in RFC 6120 and RFC 6121	2.15	Partially Met (See note 3.)
11	XMPP Extensions	2.16	Not Met. (See note 3.)
12	XML Usage	2.17	Met
13	DIFFSERV Code Point (DSCP) Requirements	2.18	Met
14	IPv6	UCR 2013, Section 5	Not Met. (See note 3.)

NOTES:

1. The annotation of "required" refers to a high-level requirement category. The applicability of each sub-requirement is provided in Enclosure 3.
2. All requirements are derived from Reference (c) except for IPv6, which is derived from Reference (d).
3. The SUT met the requirements for an XMPP Client/Server with the exceptions noted in Table 1. DISA adjudicated these exceptions as minor.
4. Security testing is accomplished by DISA-led Information Assurance test teams and the results are published in a separate report, Reference (f).

LEGEND:

C	Conditional	OSD	Office of the Secretary of Defense
CR	Capability Requirement	POAM	Plan of Action & Milestones
DIFFSERV	Differentiated Server	SASL	Simple Authentication and Security Layer
DISA	Defense Information Systems Agency	SUT	System Under Test
FR	Functional Requirement	TLS	Transport Layer Security
ID	Identification	UCR	Unified Capabilities Requirements
IM	Instant Messaging	VVoIP	Voice and Video over Internet Protocol
IPv6	Internet Protocol version 6	XMPP	Extensible Messaging and Presence Protocol
		XML	Extensible Markup Language

Table 4. UC APL Product Summary

Product Identification			
Product Name	Cisco Unified Presence Server (CUPS) with Cisco Jabber		
Software Release	Cisco Unified Presence Server (CUPS) 8.6.5 with Cisco Jabber 9.2.6		
UC Product Type(s)	XMPP		
Product Description	Cisco Unified Presence Server 8.6.5 and Cisco Jabber 9.2.6 is an XMPP Server and Client solution.		
Product Components (See note 1.)	**Component Name (See note 2.)**	**Version**	**Remarks**
Cisco Unified Presence Server (Presence/IM/Chat)	Cisco Unified Computing Systems with ESXi 5.1 **UCS-B200-M1**, **UCS-B200-M2**. (See note 3.)	Cisco Unified Presence Server (CUPS) 8.6.5	IM/P Server to facilitate exchange of Instant Messages (IM).
XMPP Client	**Cisco Jabber for Windows**	Jabber 9.2.6 Windows 7	IM/P and VVoIP
Unified Communications Manager	UCS C210-M2 (with VMware), UCS-C210-M1, and UCS-C2000M2. (See note 3.)	8.6.1	
IM Compliancy Server (site provided)	**PostgreSQL**	9.1.6	External database for compliancy logging.
Common Access Card/Single sign-on solution	**OpenAM**	9.5.5	

NOTES:
1. The detailed component and subcomponent list is provided in Enclosure 3.
2. Components bolded and underlined were tested by JITC. The other components in the family series were not tested but are also certified for joint use. JITC certifies those additional components because they utilize the same software and similar hardware and JITC analysis determined them to be functionally identical for interoperability certification purposes.
3. A comprehensive list of supported hardware configurations can be found by selecting the "Cisco Unified Communications on the Cisco Unified Computing System" link at the following URL: www.cisco.com/go/swonly.

LEGEND:

APL	Approved Products List	UC	Unified Capabilities
IM/P	Instant Messaging/Presence	VVoIP	Voice and Video over Internet Protocol
JITC	Joint Interoperability Test Command	XMPP	Extensible Messaging and Presence Protocol

4. **Test Details.** This finding is based on interoperability testing, review of the vendor's Letters of Compliance (LoC), DISA adjudication of open test discrepancy reports (TDRs), and DISA Certifying Authority (CA) Recommendation for inclusion on the UC APL. Testing was conducted at JITC's Global Information Grid Network Test Facility at Fort Huachuca, Arizona, from 26 August 2013 through 6 September 2013 using Requirements derived from reference (c) test procedures derived from Reference (e). Patches were applied and regression testing was conducted on 19 and 20 November 2013. Review of the vendor's LoC was completed on 5 September 2013. DISA adjudication of outstanding TDRs was completed on 10 December 2013. Information Assurance (IA) testing was conducted by DISA-led IA test teams and the results are published in a separate report, Reference (f). Enclosure 2 documents the test results and describes the tested network and system configurations. Enclosure 3 provides a detailed list of the interface, capability, and functional requirements.

5. **Additional Information.** JITC distributes interoperability information via the JITC Electronic Report Distribution (ERD) system, which uses Sensitive but Unclassified IP Data (formerly known as NIPRNet) e-mail. Interoperability status information is available via the JITC System Tracking Program (STP). STP is accessible by .mil/.gov users at https://stp.fhu.disa.mil/. Test reports, lessons learned, and related testing documents and references are on the JITC Joint Interoperability Tool (JIT) at https://jit.fhu.disa.mil/. Due to the sensitivity of the information, the Information Assurance Accreditation Package (IAAP) that contains the approved configuration and deployment guide must be requested directly from the Unified Capabilities Certification Office (UCCO), e-mail: disa.meade.ns.list.unified-

capabilities-certification-office@mail.mil. All associated information is available on the DISA UCCO website located at http://www.disa.mil/Services/Network-Services/UCCO.

6. **Point of Contact (POC).** The JITC point of contact is Mr. Edward Mellon, commercial telephone (520) 538-5159, DSN telephone 879-5159, FAX DSN 879-4347; e-mail address edward.a.mellon.civ@mail.mil; mailing address Joint Interoperability Test Command, ATTN: JTE (Mr. Edward Mellon) P.O. Box 12798, Fort Huachuca, AZ 85670-2798. The UCCO tracking number for the SUT is 1306703.

FOR THE COMMANDER:

3 Enclosures a/s

for RIC HARRISON
Chief
Networks/Communications and UC Portfolio

Distribution (electronic mail):
DoD CIO
Joint Staff J-6, JCS
USD(AT&L)
ISG Secretariat, DISA, JTA
U.S. Strategic Command, J665
US Navy, OPNAV N2/N6FP12
US Army, DA-OSA, CIO/G-6 ASA(ALT), SAIS-IOQ
US Air Force, A3CNN/A6CNN
US Marine Corps, MARCORSYSCOM, SIAT, A&CE Division
US Coast Guard, CG-64
DISA/TEMC
DIA, Office of the Acquisition Executive
NSG Interoperability Assessment Team
DOT&E, Netcentric Systems and Naval Warfare
Medical Health Systems, JMIS IV&V
HQUSAISEC, AMSEL-IE-IS
UCCO

ADDITIONAL REFERENCES

(c) Office of the Department of Defense Chief Information Officer, "Department of Defense Unified Capabilities (UC) Extensible Messaging and Presence Protocol (XMPP) 2013 (UC XMPP 2013)," January 2013
(d) Office of the Department of Defense Chief Information Officer, "Department of Defense Unified Capabilities Requirements 2013," 1 March 2013
(e) Joint Interoperability Test Command, "Extensible Messaging and Presence Protocol (XMPP) Test Procedures for Unified Capabilities Requirements (UCR) 2013," Draft
(f) Joint Interoperability Test Command, "Information Assurance (IA) Findings Summary for Cisco Unified Presence Server (CUPS)/Jabber for Windows CUPS 8.6.5/Jabber 9.2.6 Tracking Number 1306703," Draft

CERTIFICATION SUMMARY

1. SYSTEM AND REQUIREMENTS IDENTIFICATION. The Cisco Unified Presence Server (CUPS) 8.6.5 with Cisco Jabber 9.2.6. Table 2-1 depicts the SUT identifying information and requirements source.

Table 2-1. System and Requirements Identification

System Identification	
Sponsor	Headquarters United States Army Information Systems Engineering Command (HQUSAISEC)
Sponsor Point of Contact	Mr. Robert H. Adkins, USAISEC ELIE-ISE-ES, Building 53301, Fort Huachuca, Arizona 85613, e-mail: robert.h.adkins.civ@mail.mil
Vendor Point of Contact	Cisco Systems Global Certification Team (GCT), 7025-2 Kit Creek Road, Research Triangle Park, North Carolina 27709, e-mail: certteam@cisco.com
System Name	Cisco Unified Presence Server 8.6.5 with Cisco Jabber 9.2.6
Increment and/or Version	8.6.5/9.2.6
Product Category	XMPP Client/Server
System Background	
Previous certifications	No previous certifications
Tracking	
UCCO ID	1306703
System Tracking Program ID	4756
Requirements Source	
Unified Capabilities Requirements	Unified Capabilities XMPP 2013, UCR 2013
Remarks	
Test Organization(s)	Joint Interoperability Test Command, Fort Huachuca, Arizona
LEGEND: ID Identification UCCO Unified Capabilities Connection Office XMPP Extensible Messaging and Presence Protocol	

2. SYSTEM DESCRIPTION. The SUT is an Extensible Messaging and Presence Protocol (XMPP) Server and Client solution. The SUT is a distributed solution that connects to an Assured Services Local Area Network (ASLAN). CUPS is a highly redundant solution, which can consist of multiple sub-clusters each with up to six nodes. Cisco Jabber 9.2 provides the user with an Instant Messaging (IM) client. As an optional site add-on, a PostgreSQL 9.1 server running on RHEL 5 is deployed as Required Ancillary Equipment (RAE) to meet the IM compliancy logging and persistent chat requirements. CUPS 8.6.5 requires Cisco Unified Communications Manager (UCM) 8.6.1 as part of the solution. The SUT includes the following components.

Cisco Unified Presence Server (CUPS). CUPS provides the XMPP server to facilitate the exchange of Instant Messages. CUPS is installed as a virtual appliance on the Cisco Unified Computing System (UCS) running ESXi 5.1.

Cisco Jabber. Cisco Jabber provides the XMPP client. Cisco Jabber is installed on a Security Technical Implementation Guideline (STIG)-compliant Windows based workstation and provides the ability to send and receive IMs.

Cisco Unified Communications Manager (UCM). The SUT includes one UCM server for the CUPS/Jabber client to register.

PostgreSQL. PostgreSQL provides the external database for compliancy logging and persistent chat. Access to the database is done via PostgreSQL Admin tool installed on the management workstation.

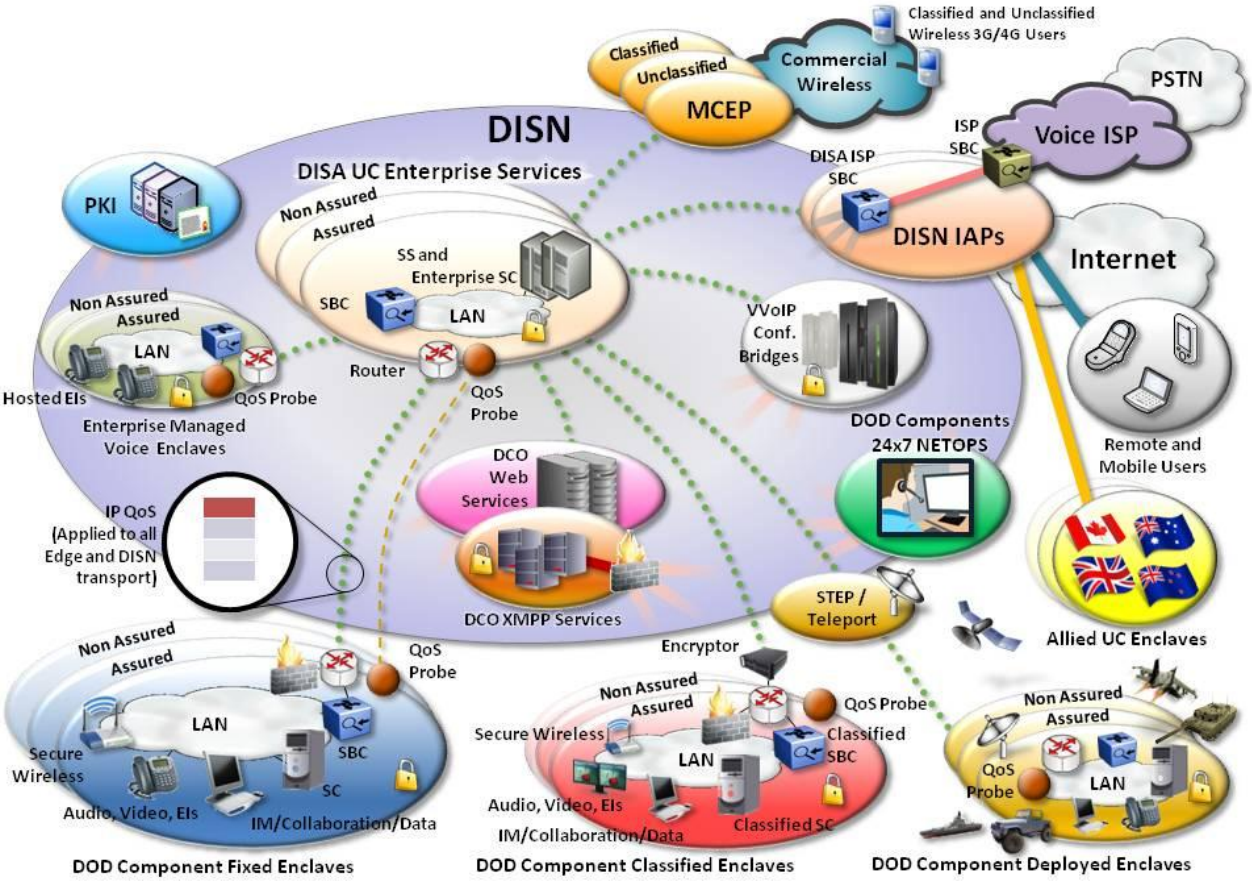
OpenAM. OpenAM provides core identity services to simplify the implementation of transparent single sign-on (SSO) as a security component in a network infrastructure. OpenAM provides the foundation for integrating diverse web applications that might typically operate against a disparate set of identity repositories and are hosted on a variety of platforms such as web and application servers.

Management Description: Cisco Unified Presence Server primary management interface is a web-based administrative console; additionally it can also be accessed via Command Line Interface (CLI). To access it, an auditor or administrative user will connect via Common Access Card (CAC) for authentication from a STIG compliant management workstation. Once connected, they would establish a management session to the Cisco Unified Presence Server management interfaces. The Cisco Unified Real-Time Monitoring Tool (RTMT), which runs as a client-side application, connects over Hypertext Transfer Protocol Secure (HTTPS) to the CUPS appliance and is used for monitoring performance, and access to the system logs. This tool does not provide a means to modify the configuration or security of the appliance.

3. OPERATIONAL ARCHITECTURE. The Unified Capabilities (UC) architecture is a two-level network hierarchy consisting of Defense Information Systems Network (DISN) backbone switches and Service/Agency installation switches. The Department of Defense (DoD) Chief Information Officer (CIO) and Joint Staff policy and subscriber mission requirements determine which type of switch can be used at a particular location. The UC architecture, therefore, consists of several categories of switches. Figure 2-1 depicts the notional operational UC architecture in which the SUT may be used.

4. TEST CONFIGURATION. The test team tested the SUT at JITC, Fort Huachuca, Arizona in a manner and configuration similar to that of a notional operational environment. Testing of the system's required functions and features was conducted using the test configuration depicted in Figure 2-2. Information Assurance (IA) testing used the same configuration.

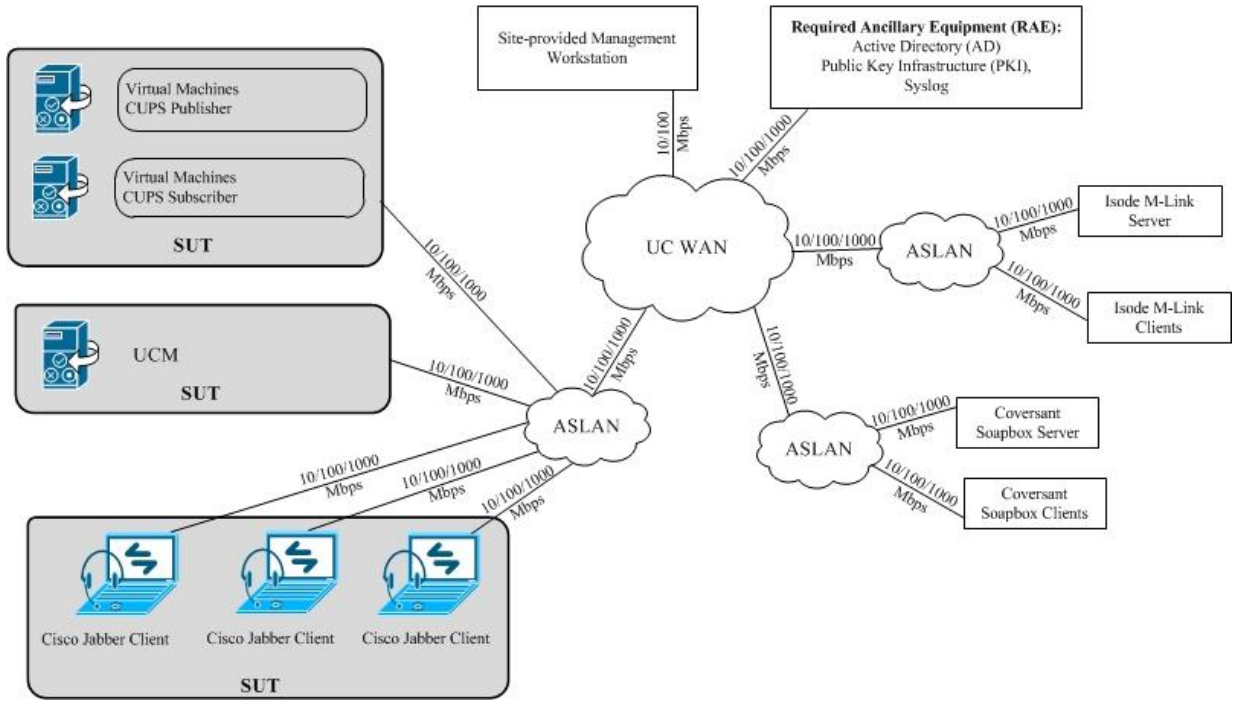
5. METHODOLOGY. Testing was conducted using XMPP requirements derived from the Unified Capabilities XMPP Requirements and the Unified Capabilities Requirements (UCR), References (c) and (d), and XMPP test procedures, Reference (e). Any discrepancies noted were written up in Test Discrepancy Reports (TDRs). The vendor submitted Plan of Action and Milestones (POA&M) as required. The remaining open TDRs were adjudicated by DISA as Minor. Any new discrepancy noted in the operational environment will be evaluated for impact on the existing certification. These discrepancies will be adjudicated to the satisfaction of DISA via a vendor POA&M, which will address all new critical TDRs within 120 days of identification.



LEGEND:

DCO	Defense Connection Online	NETOPS	Network Operations
DISA	Defense Information Systems Agency	PKI	Public Key Infrastructure
DISN	Defense Information Systems Network	PSTN	Public Switched Telephone Network
DoD	Department of Defense	QoS	Quality of Service
EI	End Instrument	SBC	Session Border Controller
IAP	Integrated Access Point	SC	Session Controller
IM	Instant Messaging	SS	Softswitch
IP	Internet Protocol	UC	Unified Capabilities
ISP	Internet Service Provider	VVoIP	Voice and Video over IP
LAN	Local Area Network	XMPP	Extensible Messaging and Presence Protocol
MCEP	Multi-Carrier Entry Point		

Figure 2-1. Notional UC Network Architecture



LEGEND:			
ASLAN	Assured Services Local Area Network	UC	Unified Capabilities
CUPS	Cisco Unified Presence Server	UCM	Unified Communications Manager
Mbps	Megabits per second	WAN	Wide Area Network
SUT	System Under Test		

Figure 2-2. SUT Test Configuration

6. **INTEROPERABILITY REQUIREMENTS, RESULTS, AND ANALYSIS.** The Capability Requirements (CR), Functional Requirements (FR), and other requirements for UC XMPP are established by DoD UC XMPP 2013, sections 2.6 through 2.18.

a. **Interfaces.** Table 3-1 provides the SUT interfaces and their testing status. There are no minimum interface requirements for an XMPP client/server system, which traverses an established Local Area Network (LAN). The SUT client and server were tested and met the requirements over a 10/100/1000 Megabits per second (Mbps) LAN. The UC XMPP specification also does not define minimum network management requirements for an XMPP Client/Server product. The SUT management requirements were successfully tested over a 10/100/1000 Mbps LAN.

b. **Capability and Functional Requirements and Status**

(1) The DoD UC XMPP 2013, section 2.6, states that an Extensible Markup Language (XML) stream provides the fundamental transport needed for all client-to-server and server-to-server communications. The ability to establish and maintain an XML stream is an essential capability of XMPP. The high-level XML stream requirements are included in the subparagraphs below.

(a) Transport Control Protocol (TCP) Binding. An initiating entity SHALL open a TCP connection to the receiving entity before it negotiates XML streams with the receiving entity. The parties then maintain that TCP connection for as long as the XML streams are in use. The SUT met this requirement with the vendor's Letter of Compliance (LoC).

(b) Stream Features

1. The initiating entity SHALL initiate an XML stream by sending an initial stream header to the receiving entity. The SUT met this requirement with the vendor's LoC.

2. In response, the receiving entity SHALL send a response stream header to the initiating entity. The SUT met this requirement with the vendor's LoC.

3. After the receiving entity has sent a response stream header to the initiating entity, the receiving entity SHALL send a <features/> child element (prefixed by the streams namespace prefix) to the initiating entity in order to announce any conditions for continuation of the stream negotiation process. Each condition takes the form of a child element of the <features/> element, qualified by a namespace that is different from the streams namespace and the content namespace. The <features/> element can contain one child, contain multiple children, or be empty. The initiating entity SHALL be capable of handling a <features/> element that contains one child or contains multiple children or that is empty. The SUT met this requirement with the vendor's LoC.

4. For stream features that are mandatory-to-negotiate, the definition of that feature SHALL declare that the feature is always mandatory-to-negotiate (e.g., this is true of resource binding for XMPP clients) or the receiving entity SHALL explicitly flag the feature as mandatory-to-negotiate (e.g., this is done for Transport Layer Security (TLS) by including an empty <required/> element in the advertisement for the STARTTLS feature). The SUT met this requirement with the vendor's LoC.

5. If the <features/> element contains at least one mandatory feature, then the initiating entity SHALL continue with the stream negotiation process. An empty <features/> element indicates that the stream negotiation is complete and that the initiating entity is cleared to send XML stanzas. The SUT met this requirement with the vendor's LoC.

(c) Stream Restarts. On successful negotiation of a feature that necessitates a stream restart, both the initiating entity and the receiving entity SHALL consider the previous stream to be replaced, but SHALL NOT terminate the underlying TCP connection; instead, the initiating entity and the receiving entity SHALL reuse the existing connection. The initiating entity then SHALL send a new initial stream header to the receiving entity. When the receiving entity receives the new initial stream header, it SHALL generate a new stream ID (instead of reusing the old stream ID) and SHALL then send a new response stream header to the initiating entity. The SUT met these requirements with the vendor's LoC.

(d) Continuation and Completion of Stream Negotiation. The receiving entity SHALL send an updated list of stream features to the initiating entity after a stream restart. The

receiving entity SHALL indicate completion of the stream negotiation process by sending to the initiating entity either an empty <features/> element or a <features/> element that contains only voluntary features. Once stream negotiation is complete, the initiating entity is cleared to send XML stanzas over the stream for as long as the stream is maintained by both parties. The SUT met these requirements with the vendor's LoC.

(e) Directionality. For client-to-server sessions, a server SHALL allow a client to use "two streams over a single TCP connection." For server-to-server sessions, the two server peers SHALL use two streams over two TCP connections, where one TCP connection is used for the stream in which stanzas are sent from the initiating entity to the receiving entity and the other TCP connection is used for the stream in which stanzas are sent from the receiving entity to the initiating entity. The SUT met these requirements with the vendor's LoC.

(f) Closing a Stream. Client and server implementations SHALL be capable of closing an XML stream by sending a closing </stream> tag. After the entity that sent the first closing stream tag receives a reciprocal closing stream tag from the other party, it SHALL terminate the underlying TCP connection or connections. The SUT met these requirements with the vendor's LoC.

(g) Stream Attributes

1. Initial Streams. For client-to-server connections, it is assumed that the client knows the associated XMPP account name of the form <localpart@domain>. The client SHALL include the "from" attribute in the initial stream header it sends to the server and SHALL set the value to the associated XMPP account name of the form <localpart@domain>. For server-to-server connections, the initiating entity SHALL include the "from" attribute in the initial stream header it sends to the receiving entity and SHALL set its value to a hostname serviced by the initiating entity. For both client-to-server and server-to-server connections, the initiating entity SHALL include the "to" attribute in the initial stream header that it sends to the receiving entity and SHALL set its value to a hostname that the initiating entity knows or expects the receiving entity to service. For both client-to-server and server-to-server connections, the initiating entity SHALL include a "version" attribute whose value is "1.0" (or higher) in the initial stream headers it generates. The SUT met these requirements with the vendor's LoC.

2. Response Streams. For both client-to-server and server-to-server connections, the receiving entity SHALL include the "from" attribute in the response stream header that it sends to the initiating entity and SHALL set its value to a hostname serviced by the receiving entity. For response stream headers in client-to-server communication, if the client included a "from" attribute in the initial stream header then the server SHALL include a "to" attribute in the response stream header and SHALL set its value to the bare JID specified in the "from" attribute of the initial stream header. If the client did not include a "from" attribute in the initial stream header then the server SHALL NOT include a "to" attribute in the response stream header. For server-to-server connections, the receiving entity SHALL include the "to" attribute in the response stream header that it sends to the initiating entity and SHALL set its value to the hostname specified in the "from" attribute of the initial stream header. For both client-to-server and server-to-server connections, the receiving entity SHALL include an "id" attribute in the response stream header that it sends to the initiating entity. The "id" attribute communicates a unique

identifier for the stream, called a STREAM ID. The stream "id" shall have the property of randomness. For both client-to-server and server-to-server connections, the receiving entity SHALL include a "version" attribute where the value is 1.0 (or higher) in the response stream headers it sends to the initiating entity. The SUT met these requirements with the vendor's LoC.

(h) Namespaces

1. Streams Namespace. Client and server implementations SHALL qualify the root <stream/> element ("stream header") by the namespace "http://etherx.jabber.org/streams" (the "streams namespace"). If this rule is violated, the entity that receives the offending stream header SHALL return a stream error to the sending entity, which SHALL be either <invalid-namespace/> or <bad-format/>. The SUT met this requirement with the vendor's LoC.

2. Content Namespace. An entity (client or server) SHALL declare a content namespace for data sent over the stream. The content namespace SHALL be the same for the initial stream and the response stream so that both streams are qualified consistently. The content namespace applies to all first-level child elements sent over the stream unless explicitly qualified by another namespace. The XMPP defines two content namespaces: "jabber:client" and "jabber:server." Client implementations SHALL support the jabber:client content namespace. Server implementations SHALL support both the jabber: client content namespace (when the stream is used for communication between a client and a server) and the jabber:server content namespace (when the stream is used for communication between two servers). If an entity receives a first-level child element qualified by a content namespace it does not support, it SHALL return an <invalid-namespace/> stream error.

(i) Stream Errors. The error child SHALL be sent by an entity (client or server) if it perceives that a stream-level error has occurred. Stream-level errors are unrecoverable. Therefore, if an error occurs at the level of the stream, the entity (client or server) that detects the error SHALL send an <error/> element with an appropriate child element that specifies the error condition and at the same time send a closing </stream> tag. The entity that generates the stream error then SHALL close the stream as explained under Section 4.4 of Request for Comments (RFC) 6120. If the error is triggered by the initial stream header, the receiving entity SHALL still send the opening <stream> tag, include the <error/> element as a child of the stream element, and then send the closing </stream> tag (preferably all at the same time). The SUT met this requirement with the vendor's LoC with the following minor exception. The SUT does not correctly respond to stream errors. Instead of responding with a stream error immediately and closing the stream, the SUT terminates the connection non-gracefully. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.

(2) The DoD UC XMPP 2013, section 2.7, states that all XML streams (i.e., including both client-to-server and server-to-server connections) SHALL be secure with the use of the TLS protocol. The SUT met this requirement with the vendor's LoC. The high-level TLS and STARTTLS Negotiation requirements are included in the subparagraphs below.

(a) STARTTLS Process. The use of the STARTTLS command to initiate TLS negotiation is mandated. All client and server implementations SHALL support and use the "STARTTLS" extension. Immediately after the opening of the response stream, the receiving entity SHALL initiate the process of stream negotiation. In the stream feature announcement provided by the receiving entity during the initial stage of the stream negotiation process, the receiving entity SHALL advertize ONLY the STARTTLS feature (qualified by the XML namespace: "urn:ietf:params:xml:ns:xmpp-tls") and SHALL also include an empty <required/> child element. The SUT met these requirements with the vendor's LoC.

(b) Initiation of STARTTLS Negotiation. In order to begin the STARTTLS negotiation, the initiating entity SHALL issue the STARTTLS command (i.e., a <starttls/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-tls' namespace) to instruct the receiving entity that it wishes to begin a STARTTLS negotiation to secure the stream. The receiving entity SHALL reply with a <proceed/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-tls' namespace. The SUT met these requirements with the vendor's LoC.

(c) STARTTLS Negotiation Fails. If there is a failure of STARTTLS negotiations, the receiving entity SHALL return a <failure/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-tls' namespace and SHALL close the XML stream. The SUT met this requirement with the vendor's LoC.

(d) TLS Negotiation. After the receiving entity has sent and the initiating entity has received the <proceed/> element, the initiating and receiving entities SHALL proceed to TLS negotiation. The TLS negotiation and implementation SHALL be in accordance with all applicable DoD Security Technical Implementation Guideline (STIG) requirements [including DoD Public Key Infrastructure (PKI) compliance] and TLS/PKI implementation/interoperability requirements as defined in Unified Capabilities Requirements (UCR) 2013, Section 4, Information Assurance. The SUT met this requirement with testing and the vendor's LoC. In addition, security testing is accomplished via DISA-led Information Assurance test teams and the results published in a separate report, Reference (f).

(e) TLS Success. If the TLS negotiation is successful, then the initiating and receiving entities SHALL proceed as in the following subparagraphs. The SUT met these requirements with testing.

1. The initiating entity SHALL send a new initial stream header to the receiving entity over the encrypted connection. The initiating entity SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the receiving entity and initiating entity MUST consider the original stream to be replaced upon success of the TLS negotiation.

2. The receiving entity SHALL respond with a new response stream header over the encrypted connection. In this new response stream header, the receiving entity SHALL generate a new stream ID instead of reusing the old stream ID.

3. The receiving entity also SHALL send stream features to the initiating entity, which SHALL NOT include the STARTTLS feature, but which SHALL advertise support of Simple Authentication and Security Layer (SASL) negotiation as described in Section 2.8, Authentication and SASL Negotiation.

(f) TLS Failure. If the TLS negotiation results in failure, the receiving entity SHALL terminate the TCP connection. The SUT met this requirement with testing.

(g) Order of TLS and SASL Negotiation. Client and server implementations SHALL complete STARTTLS negotiation before proceeding to SASL protocol negotiation; this order of negotiation is necessary to help safeguard authentication information sent during SASL negotiation, as well as to make it possible to base the use of the SASL EXTERNAL mechanism on a certificate provided during prior TLS negotiation (for entities who authenticate using a DoD PKI certificate). The SUT met this requirement with the vendor's LoC.

(h) STARTTLS Failure Case. If the STARTTLS negotiation fails, the receiving entity SHALL return a <failure/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-tls' namespace, terminate the XML stream, and terminate the underlying TCP connection. The SUT met this requirement with the vendor's LoC.

(3) The DoD UC XMPP 2013, section 2.8, states that all client and server implementations SHALL support SASL negotiations. The entities involved in an XML stream SHALL consider SASL as mandatory-to-negotiate. Anonymous login capability is prohibited. The high-level Authentication and SASL Negotiation requirements are included in the subparagraphs below. The SUT met these requirements with the vendor's LoC with any exceptions noted.

(a) Client-to-Server Streams

1. During the prior TLS negotiation, the server SHALL authenticate using a DoD PKI certificate. The client SHALL validate the certificate presented by the server.

2. The client SHALL authenticate using name and password using the SASL PLAIN mechanism (RFC 4616) as defined in the following text. As defined by this specification, the SASL PLAIN mechanism SHALL only be used when the underlying XML stream is protected using TLS. Client authentication using name and password is a minimum requirement. Client authentication using a Department of Defense (DoD) Public Key Infrastructure (PKI) certificate is preferred. The client in this scenario would comply with the behavior defined for the "initiating entity" in Section 2.8.2, Server-to-Server Streams.

3. After successful STARTTLS negotiation, the server SHALL offer the SASL PLAIN mechanism to the client during SASL negotiation. The <mechanisms/> element SHALL be qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace. The <mechanisms/> element SHALL contain one <mechanism/> child element including the appropriate value for the PLAIN mechanism.

4. The client SHALL select the PLAIN authentication mechanism by sending an <auth/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace and which SHALL include the appropriate value for the PLAIN ‘mechanism’ attribute.

5. Upon receipt of the message, the server will verify the presented authentication identity and password by performing a directory lookup to a directory service linked to the XMPP server for authenticating the user.

6. All users SHALL be linked to a directory service, which is linked to the user’s home XMPP server.

7. The server SHALL report the success of the handshake by sending a <success/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace.

8. After successful SASL negotiation, the client and server SHALL restart the stream. Upon receiving the <success/> element, the client SHALL initiate a new stream over the existing TLS connection by sending a new initial stream header to the server. The client SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the server and client MUST consider the original stream to be replaced upon sending or receiving the <success/> element.

9. Upon receiving the new initial stream header from the client, the server SHALL respond by sending a new response stream header to the client (for which it SHALL generate a new stream ID instead of re-using the old stream ID). The SUT does not generate a new Client-to-Server stream. The SUT reuses the old stream instead. DISA has accepted and approved the vendor’s POA&M and adjudicated this discrepancy as having a minor operational impact.

10. The server SHALL also send stream features, containing any further available features or containing no features (via an empty <features/> element).

(b) Server-to-Server Streams

1. During the prior TLS negotiation, the initiating entity and the receiving entity SHALL mutually authenticate using DoD PKI certificates. Server-to-server mutual authentication SHALL be in accordance with all applicable DoD STIG requirements (including DoD PKI compliance) and TLS/PKI implementation/interoperability requirements as defined in UCR 2013, Section 4, Information Assurance.

2. After the successful mutual authentication of the receiving entity and the initiating entity during the prior TLS negotiation, the receiving entity SHALL offer the SASL EXTERNAL mechanism (as defined in Appendix A of RFC 4422) to the initiating entity during SASL negotiation.

3. The receiving entity SHALL include an empty <required/> element in its advertisement of the SASL feature. The SUT does not include empty <required/> element in its

advertisement of the SASL. The SUT does not comply with Server Generation of Outbound Presence Probe. DISA has accepted and approved the vendor's POA&Ms and adjudicated these discrepancies as having a minor operational impact.

4. In response to the receiving entity offering the SASL EXTERNAL mechanism, the initiating entity SHALL select the EXTERNAL authentication mechanism by sending an <auth/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace and which SHALL include the appropriate value for the EXTERNAL 'mechanism' attribute and which also includes an empty response of "=."

5. The receiving entity SHALL report the success of the handshake by sending a <success/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace.

6. After successful SASL negotiation, the initiating entity and the receiving entity SHALL restart the stream. Upon receiving the <success/> element, the initiating entity SHALL initiate a new stream over the existing TLS connection by sending a new initial stream header to the receiving entity. The initiating entity SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the receiving entity and initiating entity MUST consider the original stream to be replaced upon sending or receiving the <success/> element.

7. Upon receiving the new initial stream header from the initiating entity, the receiving entity SHALL respond by sending a new response stream header to the initiating entity (for which it SHALL generate a new stream ID instead of reusing the old stream ID).

8. The receiving entity SHALL also send stream features, containing any further available features or containing no features (via an empty <features/> element).

(c) Simple Authentication and Security Layer (SASL) Failure. The receiving entity SHALL report failure of the handshake by sending a <failure/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace. The particular cause of failure SHALL be communicated in an appropriate child element of the <failure/> element as defined under Section 6.4 (SASL Errors) of RFC 6120. The receiving entity SHALL allow a configurable number of retries (at least two and no more than three per IM STIG policy). If the initiating entity exceeds the maximum number of retries, the server SHALL return a stream error (which SHALL be either <policy-violation/> or <not-authorized/>). The SUT does not fully comply with SASL Failure Requirements. The SUT meets SASL error conditions outlined in RFC 3920 and not RFC 6120. The SUT does not allow a configurable number of retries. DISA has accepted and approved the vendor's POA&Ms and adjudicated these discrepancies as having a minor operational impact.

(4) The DoD UC XMPP 2013, section 2.9, states that all client and server implementations SHALL support resource binding. For client-to-server connections, both the client and server SHALL consider resource binding as mandatory-to-negotiate. The SUT met these requirements and the requirements in the following subparagraphs with the vendor's LoC.

(a) Advertising Support. Upon sending a new response stream header to the client after successful SASL negotiation, the server SHALL include a <bind/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-bind' namespace in the stream features it presents to the client.

(b) Server-Generated Resource Identifier. A server implementation SHALL be able to generate an XMPP resourcepart on behalf of a client. A resourcepart SHALL at a minimum, be unique among the connected resources for a specific local account in the form of <localpart@domain>. Enforcement of this policy is the responsibility of the server. A client SHALL request a server-generated resourcepart by sending an Info/Query (IQ) stanza of type "set" (see Section 2.11.2, Roster-Related Methods) containing an empty <bind/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-bind' namespace. Once the server has generated an XMPP resourcepart for the client, it SHALL return an IQ stanza of type "result" to the client, which SHALL include a <jid/> child element that specifies the full JID for the connected resource as determined by the server.

(5) The DoD UC XMPP 2013, section 2.9, states that client and server implementations SHALL support the syntax and semantics associated with the message, presence, and IQ stanzas.

(a) All client and server implementations SHALL support resource binding.

(b) For client-to-server connections, both the client and server SHALL consider resource binding as mandatory-to-negotiate.

(c) Upon sending a new response stream header to the client after successful SASL negotiation, the server SHALL include a <bind/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-bind' namespace in the stream features it presents to the client.

(d) A server implementation SHALL be able to generate an XMPP resourcepart on behalf of a client.

(e) A resourcepart SHALL at a minimum, be unique among the connected resources for a specific local account in the form of <localpart@domain>. Enforcement of this policy is the responsibility of the server.

(f) A client SHALL request a server-generated resourcepart by sending an Info/Query (IQ) stanza of type "set" containing an empty <bind/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-bind' namespace.

(g) Once the server has generated an XMPP resourcepart for the client, it SHALL return an IQ stanza of type "result" to the client, which SHALL include a <jid/> child element that specifies the full JID for the connected resource as determined by the server.

(6) The DoD UC XMPP 2013, section 2.10, states that client and server implementations SHALL support the syntax and semantics associated with the message, presence, and IQ stanzas. The SUT met these requirements and the requirements in the following subparagraphs with testing and the vendor's LoC.

(a) Common Attributes

1. The following rules SHALL be followed regarding the use of the ‘to’ attribute in the context of XML streams qualified by the ‘jabber:client’ namespace.

a. A stanza with a specific intended recipient SHALL possess a ‘to’ attribute whose value is an XMPP address.

b. A stanza sent from a client to a server for direct processing by the server on behalf of the client (e.g., presence sent to the server for broadcasting to other entities) SHALL NOT possess a ‘to’ attribute.

2. The following rules SHALL be followed regarding the use of the ‘to’ attribute in the context of XML streams qualified by the ‘jabber:server’ namespace (i.e., server-to-server streams)

a. A stanza SHALL possess a ‘to’ attribute whose value is an XMPP address; if a server receives a stanza that does not meet this restriction, it SHALL generate an <improper-addressing/> stream error.

b. The domain identifier portion of the JID in the ‘to’ attribute SHALL match a hostname serviced by the receiving server; if a server receives a stanza that does not meet this restriction, it SHALL generate a <host-unknown/> or <host-gone/> stream error.

3. The following rules SHALL be followed regarding the use of the ‘from’ attribute in the context of XML streams qualified by the ‘jabber:client’ namespace (i.e., client-to-server streams).

a. When the server receives an XML stanza from a client, the server SHALL add a ‘from’ attribute to the stanza or override the ‘from’ attribute specified by the client, where the value of the ‘from’ attribute is the full JID (<localpart@domainpart/resource>) determined by the server for the connected resource that generated the stanza or the bare JID (<localpart@domainpart>) in the case of subscription-related presence stanzas.

b. When the server generates a stanza from the server itself for delivery to the client, the stanza SHALL include a ‘from’ attribute whose value is the bare JID (i.e., <domain>) of the server as agreed upon during stream negotiation (e.g., based on the ‘to’ attribute of the initial stream header).

c. When the server generates a stanza from the server for delivery to the client on behalf of the account of the connected client (e.g., in the context of data storage services provided by the server on behalf of the client), the stanza SHALL either (a) not include a ‘from attribute or (b) include a 'from' attribute whose value is the account's bare JID (<localpart@domainpart>).

d. A server SHALL NOT send to the client a stanza without a ‘from’ attribute if the stanza was not generated by the server (e.g., if it was generated by another client or another server).

e. When a client receives a stanza that does not include a ‘from’ attribute, it SHALL assume that the stanza is from the user’s account on the server.

4. For <iq/> stanzas, the originating entity SHALL include an ‘id’ attribute.

5. If the generated stanza includes an ‘id’ attribute, then it is required for the associated response or error stanza to also include an ‘id’ attribute, where the value of the ‘id’ attribute SHALL match that of the generated stanza.

6. If an inbound stanza received by a client or server does not possess an ‘xml:lang’ attribute, an implementation SHALL assume that the default language is that which is specified for the stream.

7. A server SHALL NOT modify or delete the ‘xml:lang’ attribute of stanzas it receives from other entities.

(b) Basic Semantics. When a client or server implementation generates or processes an IQ stanza, the following rules apply.

1. An IQ stanza SHALL include the ‘id’ attribute.

2. An IQ stanza SHALL include the ‘type’ attribute.

3. The value of the ‘type’ attribute for IQ stanzas SHALL be one of the following (if the value is other than one of the following strings, the recipient or an intermediate server SHALL return a stanza error of <bad-request/>).

a. get – The stanza requests information (e.g., the stanza inquires about data which is needed in order to complete further operations).

b. set – The stanza provides data that is needed for an operation to be completed (e.g., it sets new values, replaces existing values).

c. result – The stanza is a response to a successful “get” or “set” request.

d. error – The stanza reports an error that has occurred regarding the processing or delivery of a previously sent “get” or “set” request.

4. An entity that receives an IQ request of type “get” or “set” SHALL reply with an IQ response of type “result” or “error.” The response SHALL preserve the 'id' attribute of the request.

5. An entity that receives a stanza of type “result” or “error” SHALL NOT respond to the stanza by sending a further IQ response of type “result” or “error.”

6. An IQ stanza of type “get” or “set” SHALL contain exactly one child element, which specifies the semantics of the particular request.

7. An IQ stanza of type “result” SHALL include zero or one child element.

8. An IQ stanza of type “error” SHALL include an <error/> child.

(c) Stanza Errors. Client and server implementations SHALL comply with the mandatory requirements defined in Section 8.3 of RFC 6120.

(d) Server Rules for Processing XML Stanzas.

1. If the domainpart of the JID contained in the ‘to’ attribute does not match one of the configured hostnames of the server itself, the server SHALL attempt to route the stanza to the remote domain.

2. If a server-to-server stream already exists between the two domains, the sender’s server SHALL attempt to route the stanza to the authoritative server for the remote domain over the existing stream.

3. If no server-to-server stream exists between the two domains, the sender’s server SHALL proceed as follows: Resolve the hostname of the remote domain, Negotiate a server-to-server stream between the two domains, Route the stanza to the authoritative server for the remote domain over the newly-established stream.

4. If the routing of a stanza to the intended recipient’s server is unsuccessful, the sender’s server SHALL return an error to the sender. If resolution of the remote domain is unsuccessful, the stanza error SHALL be <remote-server-not-found/>. If the resolution succeeds, but the XML streams cannot be negotiated, the stanza error SHALL be <remote-server-timeout/>.

5. If stream negotiation with the intended recipient’s server is successful but the remote server cannot deliver the stanza to the recipient, the remote server SHALL return an appropriate error to the sender by way of the sender’s server.

6. If the hostname of the domainpart of the JID contained in the ‘to’ attribute matches one of the configured hostnames of the server, the server SHALL first determine if the hostname is serviced by the server itself or by a specialized local service. If the latter, the server SHALL route the stanza to that service. If the former, the server SHALL proceed as follows.

7. If there is no local account associated with the <localpart@domainpart>, how the stanza is processed depends on the stanza type.

a. For a message stanza, the server SHALL return a <service-unavailable/> stanza error to the sender.

b. For a presence stanza, the server SHALL ignore the stanza.

c. For an IQ stanza, the server SHALL return a <service-unavailable/> stanza error to the sender.

8. If the JID contained in the 'to' attribute is of the form <localpart@domainpart>, how the stanza is processed depends on the stanza type.

9. If the JID contained in the 'to' attribute is of the form <localpart@domainpart>, how the stanza is processed depends on the stanza type.

a. For a message stanza, if at least one connected resource for the account exists, the server SHALL deliver it to at least one of the connected resources. If there exists no connected resource, the server SHALL either return a <service-unavailable/> stanza error or store the message offline for delivery when the account next has a connected resource.

b. For a presence stanza, if at least one connected resource that has sent initial presence exists (i.e., has a "presence session"), the server SHALL deliver it to such resources. If no connected resource exists, the server SHALL ignore the stanza.

c. For an IQ stanza, the server SHALL handle it directly on behalf of the intended recipient.

10. If the JID contained in the 'to' attribute is of the form <localpart@domainpart/resource> and there is no connected resource that exactly matches the full JID, the stanza SHALL be processed as if the JID were of the form <localpart@domainpart>.

11. If the JID contained in the 'to' attribute is of the form <localpart@domainpart/resource> and there is a connected resource that exactly matches the full JID, the server SHALL deliver the stanza to that connected resource.

(7) The DoD UC XMPP 2013, section 2.11, states that in XMPP, a user's contact list is referred to as a roster. As defined in RFC 6121, a user's roster is stored by the user's server on the user's behalf so that the user can access roster information from any device. This section addresses the protocol mechanics that permit a client to retrieve a roster from its home server and to add, delete, and modify items within the roster. The SUT met these requirements and the requirements in the following subparagraphs with testing and the vendor's LoC with any exceptions noted.

(a) Roster-Related Elements and Attributes

1. Client and server implementations SHALL use IQ stanzas containing a <query/> child element qualified by the ‘jabber:iq:roster’ namespace to manage elements in a roster.

2. Client and server implementations SHALL support the ‘subscription’ attribute and the allowable subscription-related values for this attribute. The state of the presence subscription in relation to a roster item is captured in the ‘subscription’ attribute of the <item/> element. The allowable subscription-related values for this attribute are.

a. “none” – the user does not have a subscription to the contact’s presence, and the contact does not have a subscription to the user’s presence; this is the default value, so if the subscription attribute is not included, then the state is to be understood as “none.”

b. “to” – the user has a subscription to the contact’s presence, but the contact does not have a subscription to the user’s presence.

c. “from – the contact has a subscription to the user’s presence, but the user does not have a subscription to the contact’s presence.

d. “both” – both the user and the contact have subscriptions to each other’s presence (also called a “mutual subscription”).

3. In a roster result, the client SHALL ignore values of the ‘subscription’ attribute other than “none”, “to”, “from”, or “both”.

4. In a roster push, the client SHALL ignore values of the ‘subscription’ attribute other than “none”, “to”, “from”, “both”, or “remove”.

5. In a roster set, the value of the ‘subscription’ can have a value of “remove”, which indicates that the item is to be removed from the roster; the server SHALL ignore all values of the ‘subscription’ attribute other than “remove”.

6. Client implementations SHALL support the ‘name’ attribute, which is used to specify the “handle” to be associated with the JID, as determined by the user (not the contact). It is optional for a client to include the ‘name’ attribute when adding or updating a roster item.

7. Client and server implementations SHALL support the ‘ask’ attribute, which is used to specify presence subscriptions sub-state.

8. A value of “subscribe” in the ‘ask’ attribute is used to signal a “Pending Out” sub-state as described under Section 3.1.2 of RFC 6121. A server SHALL include the ‘ask’ attribute to inform the client of “Pending Out” sub-state.

9. Client and server implementations SHALL support the <group/> child element which is used to specify a category or “bucket” into which the roster item is to be

grouped by a client. It is optional for a client to include the <group/> element when adding or updating a roster item. If a roster set (Roster Set) includes no <group/> element, then the item is to be interpreted as being affiliated with no group.

(b) Roster-Related Methods

1. A client implementation SHALL have the ability to generate a Roster Get. A Roster Get is a client's request for the server to return the roster; syntactically it is an IQ stanza of type “get” sent from client to server and containing a <query/> element qualified by the ‘jabber:iq:roster’ namespace, where the <query/> element SHALL NOT contain any <item/> child elements. Likewise, a compliant server implementation SHALL be able to process this request. The expected outcome of sending a roster get is for the server to return a roster result.

2. A server implementation SHALL be able to process a Roster Get.

3. A server implementation SHALL have the ability to generate a Roster Result. A Roster Result is the server's response to a roster get; syntactically it is an IQ stanza of type “result” sent from server to client and containing a <query/> element qualified by the ‘jabber:iq:roster’ namespace. The <query/> element in a roster result contains one <item/> element for each contact and therefore can contain more than one <item/> element. The ability to generate this response is required for server implementations. Likewise, a compliant client implementation SHALL be able to process this response.

4. A client implementation SHALL be able to process a Roster Result.

5. A client implementation SHALL have the ability to generate a Roster Set.

6. A server implementation SHALL be able to process a Roster Set.

7. A server implementation SHALL have the ability to generate a Roster Push. A Roster Push is a newly created, updated, or deleted roster item that is sent from the server to the client; syntactically it is an IQ stanza of type “set” sent from server to client and containing a <query/> element qualified by the ‘jabber:iq:roster’ namespace.

8. A client implementation SHALL be able to process a Roster Push.

9. As mandated by the semantics of the IQ stanza as defined in [RFC 6120] each resource that receives a roster push SHALL reply with an IQ stanza of type ‘result’ (or ‘error’).

(c) Retrieving the Roster on Login

1. Upon authenticating with a server and binding a resource (thus becoming a connected resource), a client SHALL request the roster before sending initial presence. A client requests the roster by sending a roster get over its stream to the server.

2. The server SHALL process the roster get and SHALL return a roster result containing a <query/> element qualified by the 'jabber:iq:roster' namespace. The <query/> element in a roster result SHALL contain one <item/> element for each contact and therefore can contain more than one <item/> element.

3. If the server cannot process the roster get, it SHALL return an appropriate stanza error as described in RFC 6120.

(d) Adding a Roster Item

1. A client SHALL support the ability to add an item to the roster by sending a roster set containing a new item.

2. If the server can successfully process the roster set for the new item (i.e., if no error occurs), it SHALL create the roster item in persistent storage. The server SHALL then return an IQ stanza of type "result" to the connected resource that sent the roster set.

3. The server SHALL also send a roster push containing the new roster item to all of the user's interested resources, including the resource that generated the roster set.

4. If the server cannot successfully process the roster set, it SHALL return a stanza error.

(e) Updating a Roster Item

1. A client SHALL support the ability to update a roster item by sending a roster set to the server. Because a roster item is atomic, the item SHALL be updated exactly as provided in the roster set.

2. As with adding a roster item, if the roster item can be successfully processed, then the server SHALL update the roster information in persistent storage, send a roster push to the entire user's interested resources, and send an IQ result to the initiating resource.

(f) Deleting a Roster Item

1. A client SHALL support the ability to delete a roster item by sending a roster set and specifying the value of the 'subscription' attribute to "remove".

2. As with adding a roster item, if the server can successfully process the roster set then it SHALL update the roster information in persistent storage, send a roster push to all of the user's interested resources (with the 'subscription' attribute set to a value of 'remove'), and send an IQ result to the initiating resource.

3. The user's server SHALL generate one or more subscription-related presence stanzas, as per the following use cases. The SUT does not fully meet the Deleting a

Roster Item requirement. Presence stanza of type “unsubscribe” is not sent to contact. Instead, the stanza is silently dropped. DISA has accepted and approved the vendor’s POA&M and adjudicated this discrepancy as having a minor operational impact.

a. If the user has a presence subscription to the contact, then the user’s server SHALL send a presence stanza of type “unsubscribe” to the contact (to unsubscribe from the contact's presence).

b. If the contact has a presence subscription to the user, then the user’s server SHALL send a presence stanza of type “unsubscribed” to the contact (to cancel the contact's subscription to the user), or both.

c. If the presence subscription is mutual, then the user’s server SHALL send both a presence stanza of type “unsubscribe” and a presence stanza of type “unsubscribed” to the contact.

4. If the value of the ‘jid’ attribute specifies an item that is not in the roster, then the server SHALL return an <item-not-found/> stanza error. The SUT does not fully meet the Deleting a Roster Item requirement. Presence stanza of type “unsubscribe” is not sent to contact. Instead, the stanza is silently dropped. DISA has accepted and approved the vendor’s POA&M and adjudicated this discrepancy as having a minor operational impact.

(8) The DoD UC XMPP 2013, section 2.12, states that presence technology allows a user to subscribe to another user’s availability status and to be notified when that state changes. Before a particular user is permitted to receive information/updates regarding another user’s presence, that exchange SHALL first be authorized using a basic subscription request and approval process. When an entity receives a presence subscription request, the entity can either accept or deny the request. An entity that has a subscription to a user's presence or to which a user has a presence subscription is called a “contact.” In XMPP, a subscription lasts across presence sessions; indeed, it lasts until the contact unsubscribes or the user cancels the previously-granted subscription. In XMPP, presence subscription management is accomplished through the use of presence stanzas with specially defined attributes (“subscribe”, “unsubscribe”, “subscribed”, and “unsubscribed”). The SUT met these requirements and the requirements in the following subparagraphs with testing and the vendor’s LoC with any exceptions noted.

(a) Subscription Requests. A Subscription Request is a request from a user for authorization to permanently subscribe to a contact’s presence information; syntactically it is a presence stanza whose ‘type’ attribute has a value of “subscribe.”

1. A client implementation SHALL be capable of generating a subscription request by sending a presence stanza of type “subscribe”.

2. When the client sends a presence subscription request to a potential instant messaging and presence contact, the value of the ‘to’ attribute SHALL be a bare JID <contact@domain> rather a full JID <contact@domain/resource>.

3. Upon receiving the outbound presence subscription request, the user's server SHALL comply with the following rules for Server Processing of Outbound Subscription Requests as defined below. The SUT partially complies with Rules for Server Processing of Outbound Subscription Requests. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.

a. Before processing the request, the user's server SHALL check the syntax of the JID contained in the 'to' attribute. If the JID is of the form <localpart@domain/resourcepart> instead of <localpart@domain>, the user's server SHALL treat it as if the request had been directed to the contact's bare JID and modify the 'to' address accordingly.

b. If the potential contact is hosted on the same server as the user, then the server SHALL adhere to the Rules for Server Processing of Inbound Subscription Requests (see below) and SHALL deliver it to the local contact.

c. If the potential contact is hosted on a remote server, the user's server SHALL then route the stanza to that remote domain in accordance with the Server Rules for Processing XML Stanzas.

4. When a server processes or generates an outbound presence stanza of type "subscribe", "subscribed", "unsubscribe", or "unsubscribed", the server SHALL stamp the outgoing presence stanza with the bare JID <localpart@domain> of the sending entity. Enforcement of this rule simplifies the presence subscription model and helps to prevent presence leaks.

5. If the presence subscription request cannot be locally delivered or remotely routed (e.g., because the request is malformed, the local contact does not exist, the remote server does not exist, an attempt to contact the remote server times out, or any other error determined or experienced by the user's server), then the user's server SHALL return an appropriate error stanza to the user.

6. After locally delivering or remotely routing the presence subscription request, the user's server SHALL then send a roster push to all of the user's interested resources, containing the potential contact with a subscription state of "none" and with notation that the subscription is pending (via an 'ask' attribute whose value is "subscribe").

(b) Cancelling a Subscription

1. A client implementation SHALL be capable of sending a presence stanza of type "unsubscribed" in order to cancel a subscription that it has previously granted to a user.

2. Upon receiving the outbound subscription cancellation, the contact's server SHALL proceed as in the following subparagraphs. The SUT met these requirements through testing with the following minor exception. The SUT partially complies to Rules for Server Processing of Outbound Subscription Cancellation. Upon receiving the outbound subscription cancellation, the contact's server does not send a presence stanza of type "unavailable" from all

of the contacts online resources to the user. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.

a. If the user is hosted on the same server as the contact, then the server SHALL adhere to the rules specified in the next section in processing the subscription cancellation.

b. If the user is hosted on a remote server, the contact's server SHALL then route the stanza to that remote domain.

c. As mentioned, before locally delivering or remotely routing the stanza, the contact's server SHALL stamp the outbound subscription cancellation with the bare JID <localpart@domain> of the contact.

d. The contact's server then SHALL send a roster push with the updated roster item to all of the contact's interested resources, where the subscription state is now either "none" or "to."

e. The contact's server then SHALL send a presence stanza of type "unavailable" from all of the contact's online resources to the user.

3. When the user's server receives the inbound subscription cancellation, it SHALL first check if the contact is in the user's roster with subscription='to' or subscription='both'.

a. If this check is successful, the user's server SHALL deliver the inbound subscription cancellation to all of the user's interested resources. This SHALL occur before sending the roster push described in the next step.

b. Initiate a roster push to all of the user's interested resources, containing an updated roster item for the contact with the 'subscription' attribute set to a value of "none" (if the subscription state was "To" or "To + Pending In") or "from" (if the subscription state was "Both").

c. If the check (above) is not successful, that is, if the user does not exist, if the contact is not in the user's roster, or if the contact is in the user's roster with a subscription state other than those described in the foregoing check, then the user's server SHALL silently ignore the stanza by not delivering it to the user, not modifying the user's roster, and not generating a roster push to the user's interested resources.

4. To unsubscribe from a contact's presence, the client SHALL send a presence stanza of type "unsubscribe".

5. Upon receiving the outbound unsubscribe, the user's server SHALL proceed as follows.

a. If the contact is hosted on the same server as the user, then the server SHALL adhere to the rules specified for Server Processing of Inbound Unsubscribe.

b. If the contact is hosted on a remote server, the user's server SHALL then route the stanza to that remote domain.

c. The user's server then SHALL send a roster push with the updated roster item to all the user's interested resources, where the subscription state is now either "none" or "from".

6. When the contact's server receives the unsubscribe notification, it SHALL first check if the user is in the contact's roster with subscription='from' or subscription='both' (i.e., a subscription state of "From", "From + Pending Out", or "Both".

a. If this check is successful, the contact's server SHALL do the following. Deliver the inbound unsubscribe to all of the contact's interested resources. This SHALL occur before sending the roster push described in the next step. Initiate a roster push to all of the contact's interested resources, containing an updated roster item for the contact with the 'subscription' attribute set to a value of "none" (if the subscription state was "From" or "From + Pending Out") or "to" (if the subscription state was "Both"). Generate an outbound presence stanza of type "unavailable" from each of the contact's available resources to the user.

b. If the check (above) is not successful, that is, if the contact does not exist, if the user is not in the contact's roster, or if the user is in the contact's roster with a subscription state other than those described in the foregoing check, then the contact's server SHALL silently ignore the stanza by not delivering it to the contact, not modifying the contact's roster, and not generating a roster push to the contact's interested resources.

(9) The DoD UC XMPP 2013, section 2.13, states that in XMPP, presence information is exchanged using <presence/> stanzas as defined in RFC 6121. A client controlled by a user sends presence information to its home server and the home server in turn propagates that information to all of the user's contacts who have a subscription to that user's presence.

(a) Initial Presence

1. After completing the mandatory-to-negotiate stream features and retrieving a roster, a client implementation SHALL signal its availability for communication by sending initial presence to its server, i.e., a presence stanza with no 'to' address and no 'type' attribute.

2. Upon receiving initial presence from a client, the user's server SHALL send the initial presence stanza from the full JID <user@domain/resource> of the user to all contacts that are subscribed to the user's presence.

3. The user's server SHALL also broadcast initial presence from the user's newly available resource to all of the user's available resources (including the resource that generated the presence notification in the first place).

4. In the absence of presence information about the user's contacts, the user's server SHALL also send presence probes to the user's contacts on behalf of the user.

5. Upon receiving presence from the user, the contact's server SHALL deliver the user's presence stanza to all of the contact's available resources.

6. When the contact's client receives presence from the user, it SHALL proceed as follows. If the user is in the contact's roster, the client SHALL display the presence information in an appropriate roster interface. If the user is not in the contact's roster, the client SHALL ignore the presence information and not display it to the contact.

(b) Presence Probes. A presence probe is a request for a contact's current presence information, sent on behalf of a user by the user's server; syntactically it is a presence stanza whose 'type' attribute has a value of "probe." In the context of presence subscriptions, the value of the 'from' address SHALL be the bare JID of the subscribed user and the value of the 'to' address SHALL be the bare JID of the contact to which the user is subscribed, since presence subscriptions are based on the bare JID.

1. To discover the availability of a user's contact, the user's server SHALL be capable of sending a presence probe from the bare JID <user@domain> of the user to the bare JID <contact@domain> of the contact.

2. The server SHALL NOT send a probe to a contact if the user is not subscribed to the contact's presence (i.e., if the contact is not in the user's roster with the 'subscription' attribute set to a value of "to" or "both").

3. Upon receiving a presence probe to the contact's bare JID from the user's server on behalf of the user, the contact's server SHALL reply as follows.

a. If the contact account does not exist or the user is in the contact's roster with a subscription state other than "From", "From + Pending Out", or "Both" (as defined under Appendix A of RFC 6121), then the contact's server SHALL return a presence stanza of type "unsubscribed" in response to the presence probe. Here the 'from' address SHALL be the bare JID of the contact, since specifying a full JID would constitute a presence leak as described in RFC 6120.

b. Else, if the contact has no available resources, then the server SHALL reply to the presence probe by sending to the user a presence stanza of type "unavailable."

c. Else, if the contact has at least one available resource, then the server SHALL reply to the presence probe by sending to the user the full XML of the last presence stanza with no 'to' attribute received by the server from each of the contact's available resources. Here the 'from' addresses are the full JIDs of each available resource.

(c) Subsequent Presence Broadcasts

1. After sending initial presence, a client implementation SHALL be capable of updating its availability by sending a presence stanza with no ‘to’ address and no ‘type’ attribute.

2. Upon receiving a presence stanza expressing updated availability, the user’s server SHALL broadcast the full XML of that presence stanza to the contacts who meet all of the following criteria.

a. The contact is in the user’s roster with a subscription type of “from” or “both.”

b. The last presence stanza received from the contact during the user’s presence session was NOT of type “unsubscribe.”

3. The user’s server SHALL also send the presence stanza to all of the user’s available resources (including the resource that generated the presence notification in the first place).

4. Upon receiving presence from the user, the contact’s server SHALL deliver the user’s presence stanza to all of the contact’s available resources.

5. From the perspective of the contact’s client, there is no significant difference between initial presence broadcast and subsequent presence broadcast, so the contact’s client SHALL follow the rules for processing of inbound presence defined under Section 2.13.1.4, Client Processing of Inbound Initial Presence.

(d) Unavailable Presence

1. Before ending its presence session with a server, the user’s client SHALL gracefully become unavailable by sending unavailable presence, i.e., a presence stanza that possesses no ‘to’ attribute and that possesses a ‘type’ attribute whose value is “unavailable.” The unavailable presence stanza SHALL NOT contain the <priority/> element or the <show/> element, since these elements apply only to available resources.

2. The user’s server SHALL NOT depend on receiving unavailable presence from an available resource, since the resource can become unavailable ungracefully (e.g., the resource can be timed out by the server because of inactivity).

3. If an available resource becomes unavailable for any reason (either gracefully or ungracefully), the user’s server SHALL broadcast unavailable presence to all contacts that meet all of the following criteria.

a. The contact is in the user’s roster with a subscription type of “from” or “both.”

b. The last presence stanza received from the contact during the user's presence session was not of type "error" or "unsubscribe."

4. If the unavailable notification was gracefully received from the client, then the server SHALL broadcast the full XML of the presence stanza.

5. The user's server SHALL also send the unavailable notification to all of the user's available resources (including the resource that generated the presence notification in the first place).

6. If the server detects that the user has gone offline ungracefully, then the server SHALL generate the unavailable presence broadcast on the user's behalf.

7. Upon receiving an unavailable notification from the user, the contact's server SHALL deliver the user's presence stanza to all of the contact's available resources.

8. From the perspective of the contact's client, there is no significant difference between initial presence broadcast and unavailable presence broadcast, so the contact's client SHALL follow the rules for processing of inbound presence defined under Section 2.13.1.4, Client Processing of Inbound Initial Presence.

(e) Presence Syntax. To specify a particular availability sub-state, a client implementation SHALL support the <show/> element within a presence stanza. A presence stanza SHALL NOT contain more than one <show/> element. The XML character data of the <show/> element is not human-readable. The XML character data SHALL be one of the following.

1. away – The entity or resource is temporarily away.

2. chat – The entity or resource is actively interested in chatting.

3. dnd – The entity or resource is busy (dnd = "Do Not Disturb").

4. xa – The entity or resource is away for an extended period (xa = "eXtended Away").

(10) The DoD UC XMPP 2013, section 2.14, states that after a client has established and secured a stream with its home server, the next step, as discussed above, is to bind a specific resource to the stream. O nce the client has completed the resource binding step, the client may generate and exchange an unlimited number of stanzas. One such stanza that can be exchanged is <message/>. As discussed in RFC 6121, a <message/> stanza is used to "push" information to another entity.

(a) One-to-One Chat Sessions. One-to-One Chat permits a user to engage in a near real-time, text-based conversation with another user. In XMPP, this text-based conversation is enabled through the exchange of <message/> stanzas. As discussed in Section 5 of RFC 6121,

the two parties will typically exchange a number of messages in relatively rapid succession within a relatively brief period.

1. When a user's client is engaged in a chat session with a contact, the user's client SHALL send a message of type "chat" and the contact's client SHALL preserve that message type in subsequent replies.

2. The user's client SHALL be capable of including a <thread/> element with its initial message, which the contact's client SHALL also preserve during the life of the chat session. The primary use of the XMPP <thread/> element is to uniquely identify a conversation thread or "chat session" between two entities instantiated by <message/> stanzas of type 'chat'.

3. The user's client SHALL address the initial message in a chat session to the bare JID of the contact (i.e., <contact@domain>). Until and unless the user's client receives a reply from the contact, it SHALL continue sending any further messages to the contact's bare JID. Once the user's client receives a reply from the contact's full JID, it SHALL address its subsequent messages to the contact's full JID as provided in the 'from' address of the contact's replies.

4. The contact's client SHALL address its subsequent replies to the user's full JID <user@domain/resource> as provided in the 'from' address of the initial message.

(b) Message Stanza Syntax

1. An instant messaging client SHALL specify the intended recipient for a message stanza by providing the JID of the intended recipient in the 'to' attribute of the <message/> stanza.

2. An instant messaging client SHALL support all of the following message types.

a. chat. The value "chat" indicates that the message is sent in the context of a one-to-one chat session. Typically, a receiving client will present/display messages of type "chat" in an interface that enables one-to-one chat between the two parties, including an appropriate conversation history.

b. error. The value "error" indicates that the message is generated by an entity that experienced an error in processing a message received from another entity.

c. groupchat. The value "groupchat" indicates that the message is sent in the context of a multiuser chat environment. Typically, a receiving client will present a message of type "groupchat" in an interface that enables many-to-many chat between the parties.

d. normal. The value "normal" indicates that the message is a standalone message that is sent outside the context of a one-to-one conversation or groupchat, and to which it is expected that the recipient will reply. Typically, a receiving client will present a message of

type “normal” in an interface that enables the recipient to reply, but without a conversation history. The default value of the ‘type’ attribute is "normal."

e. headline. The value “headline” indicates that the message provides an alert, a notification, or other information to which no reply is expected (e.g., news headlines, sports updates, near-real-time market data, and syndicated content). Because no reply to the message is expected, typically a receiving client will present a message of type “headline” in an interface that appropriately differentiates the message from standalone messages, chat messages, or groupchat messages (e.g., by not providing the recipient with the ability to reply).

3. If an application receives a message with no ‘type’ attribute or the application does not understand the value of the ‘type’ attribute provided, it SHALL consider the message to be of type “normal”.

4. A client SHALL be capable of populating a <message/> stanza with the <body/> element. The <body/> element contains human-readable XML character data that specifies the textual content of the message.

(11) The DoD UC XMPP 2013, section 2.15, states that Section 15 of RFC 6120 and Section 13 of RFC 6121 describe a protocol feature set that summarizes the conformance requirements associated with these two specifications. In the event of a discrepancy between Section 15 of RFC 6121 or Section 13 of RFC 6121 and the UC XMPP 2013 Specification, the explicit requirements defined in the UC XMPP 2013 Specification take precedence. The SUT met these requirements with the following minor exception. The SUT establishes SASL external authentication with incorrect domain. The requirement states both the “from” field and the authentication fields should be used but are not required. RFC 6120 states the following: "Security Warning: Because it is possible for a third party to tamper with information that is sent over the stream before a security layer such as TLS is successfully negotiated, it is advisable for the receiving server to treat any such unprotected information with caution; this applies especially to the 'from' and 'to' addresses on the first initial stream header sent by the initiating entity." Cisco does not make use of the “from” field due to the security warning in RFC6120 as noted above. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.

(12) The DoD UC XMPP 2013, section 2.16, states the protocol specifications referenced within Table 2.16-1, DoD XMPP Protocol Suite, constitute a mandatory protocol suite (i.e., for the purpose of compliance testing and certification; support for these extensions is defined as REQUIRED). Where there may be some degree of ambiguity in a commercial standard regarding whether or not support for a particular capability or feature is REQUIRED, Table 2.16-2, Elevated/Clarified Requirements, adds explicit clarification. The SUT does not comply with the requirements in XMPP Extension Protocols (XEP)-0045: Multi-User Chat. DISA has accepted and approved the vendor’s POA&M and adjudicated this discrepancy as having a minor operational impact.

(13) The DoD UC XMPP 2013, section 2.17, states that XMPP client and server implementations SHALL comply with the mandatory requirements defined in Section 11 of RFC 6120. The SUT met this requirement with the vendor's LoC.

(14) The DoD UC XMPP 2013, section 2.18, states that XMPP client and server implementations shall class mark XMPP traffic consistent with the code point value defined for ROUTINE Low-Latency Data as per the DSCP Assignments defined in Section 6 of UCR 2013. The SUT met this requirement with testing and the vendor's LoC.

(15) The UCR 2013, section 5, states that the XMPP Server/Client must be IPv6 capable using the guidance in Table 5-2.4 for a Network Appliance/Simple Server (NA/SS). The SUT does not support IPv6. The Office of the Secretary of Defense (OSD) issued a waiver for the IPv6 requirements.

c. Hardware/Software/Firmware Version Identification. Table 3-3 provides the SUT components' hardware, software, and firmware tested. The JITC tested the SUT in an operationally realistic environment to determine its interoperability capability with associated network devices and network traffic. Table 3-4 provides the hardware, software, and firmware of the components used in the test infrastructure.

7. TESTING LIMITATIONS. JITC test teams noted the following testing limitations including the impact they may have on interpretation of the results and conclusions. JITC does not currently have the capabilities to test SNMPv3 however; the vendor met the requirements for this via LoC.

8. CONCLUSION(S). The SUT meets the critical interoperability requirements in accordance with UC XMPP 2013 and is certified for joint use with other UC Products listed on the Approved Products List (APL). The SUT meets the interoperability requirements for the interfaces listed in Table 3-1.

DATA TABLES

Table 3-1. Interface Status

Interface (See note 1.)	Threshold CR/FR Requirements (See note 2.)	Status	Remarks
Network Management Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1	Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1	Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1	Met	See note 3.
Server Network Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1, 2, 3	Partially Met	See note 3.
Client Interfaces			
IEEE 802.3i (10BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3u (100BaseT UTP) (C)	1, 2, 3	Partially Met	See note 3.
IEEE 802.3ab (1000BaseX) (C)	1, 2, 3	Partially Met	See note 3.

NOTES:
1. References (c) and (d) do not specify a minimum required Ethernet interface, therefore, any one of the listed interfaces can be supported.
2. The SUT high-level CR and FR ID numbers depicted in the Threshold CRs/FRs column can be cross-referenced in Table 3. These high-level CR/FR requirements refer to a detailed list of requirements provided in Enclosure 3.
3. The SUT does not support IPv6. The Office of the Secretary of Defense (OSD) granted a waiver for IPv6 on 16 May 2013.

LEGEND:

802.3ab	1000BaseT Gbps Ethernet over twisted pair at 1 Gbps (125 Mbps)	FR	Functional Requirement
		ID	Identification
802.3i	10BaseT Mbps over twisted pair	IEEE	Institute of Electrical and Electronics Engineers
802.3u	Standard for 100 Mbps Ethernet	IPv6	Internet Protocol version 6
C	Conditional	SUT	System Under Test
CR	Capability Requirement	UTP	Unshielded Twisted Pair

Table 3-2. Capability and Functional Requirements and Status

CR/FR ID	Capability/Function	Applicability (See note 1.)	UC XMPP Reference	Status
1	**XML Streams**			
	TCP Bindings	Required	2.6.1	Met
	Stream Features	Required	2.6.3	Met
	Stream Restarts	Required	2.6.4	Met
	Continuation and Completion of Stream Negotiations	Required	2.6.5	Met
	Directionality	Required	2.6.6	Met
	Closing a Stream	Required	2.6.7	Met
	Stream Attributes	Required	2.6.8	Met
	Namespaces	Required	2.6.9	Met
	Stream Errors	Required	2.6.10	Partially Met (See note 2.)
2	**TLS and STARTTLS Negotiation**			
	STARTTLS Process	Required	2.7.1	Met
	Initiation of STARTTLS Negotiation	Required	2.7.2	Met
	STARTLLS Negotiation Fails	Required	2.7.3	Met
	TLS Negotiation	Required	2.7.4	Met (See note 3.)
	TLS Success	Required	2.7.5	Met
	TLS Failure	Required	2.7.6	Met
	Order of TLS and SASL Negotiation	Required	2.7.7	Met
	STARTTLS Failure Case	Required	2.7.8	Met
3	**Authentication and SASL Negotiation**			
	Client-to-Server Streams	Required	2.8.1	Partially Met (See note 4.)
	Server-to-Server Streams	Required	2.8.2	Partially Met (See note 5.)
	SASL Failure	Required	2.8.3	Partially Met (See note 6.)

Table 3-2. Capability and Functional Requirements and Status (continued)

CR/FR ID	Capability/Function	Applicability (See note 1.)	UC XMPP Reference	Status
4	**Resource Binding**			
	Resource Binding Process	Required	2.9.2	Met
5	**XML Stanzas**			
	Common Attributes	Required	2.10.1	Met
	Basic Semantics	Required	2.10.2	Met
	Stanza Errors	Required	2.10.3	Met
	Server Rules for Processing XML Stanzas	Required	2.10.4	Met
6	**Roster Management**			
	Roster-Related Elements and Attributes	Required	2.11.1	Met
	Roster-Related Methods	Required	2.11.2	Met
	Retrieving the Roster Login	Required	2.11.3	Met
	Adding a Roster Item	Required	2.11.4	Met
	Updating a Roster Item	Required	2.11.5	Met
	Deleting a Roster Item	Required	2.11.6	Partially Met (See note 7.)
7	**Presence Subscription Management**			
	Subscription Requests	Required	2.12.1	Partially Met (See note 8.)
	Cancelling a Subscription	Required	2.12.2	Partially Met (See note 9.)
	Unsubscribing	Required	2.12.3	Partially Met (See note 10.)
8	**Exchanging Presence Information**			
	Initial Process	Required	2.13.1	Met
	Presence Probes	Required	2.13.2	Partially Met (See note 11.)
	Subsequent Presence Broadcasts	Required	2.13.3	Met
	Unavailable Presence	Required	2.13.4	Met
	Presence Syntax	Required	2.13.5	Met
9	**Exchanging Messaging**			
	One-to-One Chat Sessions	Required	2.14.1	Met
	Message to Stanza Syntax	Required	2.14.2	Met
10	**Conformance Requirements in RFC 6120 and RFC 6121**			
	Conformance Requirements in RFC 6120 and RFC 6121	Required	2.15	Partially Met (See note 12.)
11	**XMPP Extensions**			
	XMPP Extensions	Required	2.16	Not Met (See note 13.)
12	**XML Usage**			
	XML Usage	Required	2.17	Met
13	**DIFFSERV Code Point (DSCP) Requirements**			
	DSCP Requirements	Required	2.18	Met
14	**IPv6**			
	IPv6	Required	5	Not Met (See note 14.)

NOTES:

1. The annotation of 'required' refers to a high-level requirement category. The applicability of each sub-requirement is provided in Table 3-5. All requirements are derived from Reference (c) except for IPv6, which is derived from Reference (d).
2. The SUT does not correctly respond to stream errors. Instead of responding with a stream error immediately and closing the stream, the SUT terminates the connection non-gracefully. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
3. Security testing is accomplished by DISA-led Information Assurance test teams and the results published in a separate report, Reference (e).
4. The SUT does not generate a new Client-to-Server stream. The SUT reuses the old stream instead. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
5. The SUT does not include empty <required/> element in its advertisement of the SASL. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
6. The SUT does not fully comply with SASL failure requirements. The SUT meets SASL error conditions outlined in RFC 3920 and not RFC 6120. The SUT does not allow a configurable number of retries. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
7. The SUT does not fully meet the Deleting a Roster Item requirement. Presence stanza of type "unsubscribe" is not sent to contact. Instead, the stanza is silently dropped.

Table 3-2. Capability and Functional Requirements and Status (continued)

NOTES (continued):

8. The SUT partially complies with Rules for Server Processing of Outbound Subscription Requests. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
9. The SUT partially complies to Rules for Server Processing of Outbound Subscription Cancellation. Upon receiving the outbound subscription cancellation, the contact's server does not send a presence stanza of type "unavailable" from all of the contacts online resources to the user. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.
10. The SUT partially complies with Rules for Server Processing of Inbound Unsubscribe. The SUT doesn't check if the user is in the contact's roster with subscription='from' or subscription='both'. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.
11. The SUT does not comply with Server Generation of Outbound Presence Probe. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
12. The SUT establishes SASL external authentication with incorrect domain. DISA adjudicated this discrepancy as minor and stated the intent to change this requirement.
13. The SUT does not comply with the requirements in XMPP Extension XEP-0045: Multi-User Chat. DISA has accepted and approved the vendor's POA&M and adjudicated this discrepancy as having a minor operational impact.
14. The SUT does not support IPv6. The Office of the Secretary of Defense (OSD) granted a waiver for IPv6 on 16 May 2013.

LEGEND:

DISA	Defense Information Systems Agency	SASL	Simple Authentication and Security Layer
IPv6	Internet Protocol version 6	SUT	System Under Test
RFC	Request for Comments	UCR	Unified Capabilities Requirements
POA&M	Plan of Action & Milestones	XMPP	Extensible Messaging and Presence Protocol

Table 3-3. SUT Hardware/Software/Firmware Version Identification

Component (See note 1.)	Release	Sub-component	Function
Cisco Unified Computing Systems with ESXi 5.1 **UCS-B200-M1, UCS-B200-M2**. (See note 2.)	Cisco Unified Presence Server (CUPS) 8.6.5	Not Applicable	Cisco Unified Presence Server (Presence/IM/Chat)
Cisco Jabber for Windows	Jabber 9.2.6 Windows 7	Not Applicable	XMPP Client
UCS C210-M2 (with VMware), UCS-C210-M1, and UCS-C2000M2. (See note 2.)	8.6.1	Not Applicable	
PostgreSQL	9.1.6	Not Applicable	IM Compliancy Server (site provided)
OpenAM	9.5.5	Not Applicable	Common Access Card/Single sign-on solution

NOTES:

1. Components bolded and underlined were tested by JITC. The other components in the family series were not tested but are also certified for joint use. JITC certifies those additional components because they utilize the same software and similar hardware and JITC analysis determined them to be functionally identical for interoperability certification purposes.
2. A comprehensive list of supported hardware configurations can be found by selecting the "Cisco Unified Communications on the Cisco Unified Computing System" link at the following URL: www.cisco.com/go/swonly.

LEGEND:

APL	Approved Products List	UC	Unified Capabilities
IM/P	Instant Messaging/Presence	VVoIP	Voice and Video over Internet Protocol
JITC	Joint Interoperability Test Command	XMPP	Extensible Messaging and Presence Protocol

Table 3-4. Test Infrastructure Hardware/Software/Firmware Version Identification

System Name	Software Release	Function
Required Ancillary Equipment (Site provided)		
Active Directory		
Public Key Infrastructure		
SysLog Server		
Management Workstation with Microsoft Windows 7		
Test Network Components		
Isode M-Link	R15.1v5-1	XMPP Server
Isode M-Link Swift client	Swift Client 2.0D1	XMPP Client
Coversant SoapBox Server	Release 4.3	XMPP Server
Coversant SoapBox Client	Release 4.3	XMPP Client
LEGEND: XMPP Extensible Messaging and Presence Protocol		

Table 3-5. XMPP Capability/Functional Requirements

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
1	**2.6 - XML Streams**				
1-1	As XMPP is defined in this specification, an initiating entity SHALL open a TCP connection to the receiving entity before it negotiates XML streams with the receiving entity. The parties then maintain that TCP connection for as long as the XML streams are in use (Section 3.1, RFC 6120).	2.6.1 IM-000010	R	R	L
1-2	When a server receives a stanza and the JID contained in the "to" attribute does not match one of the configured hostnames of the server itself, the server SHALL attempt to route the stanza to the remote domain. If no server-to-server stream exists between the two domains, the sender's server SHALL attempt to resolve the remote hostname using a Domain Name Service (DNS) Service record query (DNS SRV query) of "xmpp-server" (for server-to-server connections) (Section 10.4 of RFC 6120).	2.6.1.1 IM-000020	R	R	L
1-3	To discover the hostname of the XMPP service in a given domain, XMPP clients SHALL use the same hostname resolution process. However, the Service identified in the DNS SRV query will be "xmpp-client" (for client-to-server connections).	2.6.1.1 IM-000030	R		L
1-4	All server and client implementations SHALL support this hostname resolution process as follows (Section 3.2.1, RFC 6120): a. The initiating entity SHALL construct a DNS SRV query (see RFC 2782) where inputs are as follows: i. A service of "xmpp-server" for server-to-server connections (or alternatively, "xmpp-client" for client-to-server connections). ii. A proto of "tcp." iii. A name corresponding to the "origin domain" of the XMPP service to which the initiating entity wishes to connect (e.g., "example.disn.mil"). b. The result is a query such as "_xmpp-server._tcp.example.disn.mil." (or alternatively, "_xmpp-client._tcp.exmple.disn.mil." for client-to-server connections). c. If a response is received, it will contain one or more combinations of a port and hostname, each of which is weighted and prioritized as described in RFC 2782. d. The initiating entity SHALL choose one of the returned hostnames to resolve (following the rules in RFC 2782), which it SHALL do by using a DNS "A" or "AAAA" lookup on the hostname; this will result in an IPv4 or IPv6 address. e. The initiating entity SHALL use the Internet Protocol (IP) address from the first successfully resolved hostname (with the corresponding port number returned by the SRV lookup) as the connection address for the receiving entity. f. If the initiating entity fails to connect using that IP address, but the "A" or "AAAA" lookup returned more than one IP address, then the initiating entity SHALL use the next resolved IP address for that hostname as the connection address. g. If the initiating entity fails to connect using all resolved IP addresses for a given hostname, then it repeats the process of resolution and connection for the next hostname returned by the SRV lookup. h. If the initiating entity fails to connect using any hostname returned by the SRV lookup, then it either SHALL abort the connection attempt or SHALL use the fallback process described in the following section. h. If the initiating entity fails to connect using any hostname returned by the SRV lookup, then it either SHALL abort the connection attempt or SHALL use the fallback process described in the following section.	2.6.1.1 IM-000040	R	R	L
1-5	The fallback process SHALL be a normal "A" or "AAAA" address record resolution to determine the IPv4 or IPv6 address of the origin domain, where the port used is the "xmpp-client" port of 5222 for client-to-server connections or the "xmpp-server" port 5269 for server-to-server connections. [Section 3.2.2, RFC 6120]	2.6.1.3 IM-000050	R	R	L
1-6	The initiating entity SHALL initiate an XML stream by sending an initial stream header to the receiving entity. C: <stream:stream from='john@im.example1.dod.mil' to='im.example1.dod.mil' version='1.0' xml:lang='en' xmlns='jabber: client' xmlns:stream='http://etherx.jabber.org/streams'>	2.6.3 IM-000060	R	R	L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
1-7	In response, the receiving entity SHALL send a response stream header to the initiating entity. (Section 5.4.1, RFC 6120) S: <stream:stream from='im.example1.dod.mil' id='t7AMCin9zjMNwQKDnplntZPIDEI=' to='john@im.example1.dod.mil' version='1.0' xml:lang='en' xmlns='jabber: client' xmlns:stream='http://etherx.jabber.org/streams'	2.6.3 IM-000070	R	R	L
1-8	After the receiving entity has sent a response stream header to the initiating entity, the receiving entity SHALL send a <features/> child element (prefixed by the streams namespace prefix) to the initiating entity in order to announce any conditions for continuation of the stream negotiation process. Each condition takes the form of a child element of the <features/> element, qualified by a namespace that is different from the streams namespace and the content namespace. The <features/> element can contain one child, contain multiple children, or be empty [Section 4.2.2, RFC 6120]. The initiating entity SHALL be capable of handling a <features/> element that contains one child or contains multiple children or that is empty.	2.6.3 IM-000080	R	R	L
1-9	For stream features that are mandatory-to-negotiate, the definition of that feature SHALL declare that the feature is always mandatory-to-negotiate (e.g., this is true of resource binding for XMPP clients) or the receiving entity SHALL explicitly flag the feature as mandatory-to-negotiate (e.g., this is done for TLS by including an empty <required/> element in the advertisement for the STARTTLS feature). [Section 4.2.2, RFC 6120] R: <stream: features> <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'> <required/> </starttls> </stream: features>	2.6.3 IM-000090	R	R	L
1-10	If the <features/> element contains at least one mandatory feature, then the initiating entity SHALL continue with the stream negotiation process. An empty <features/> element indicates that the stream negotiation is complete and that the initiating entity is cleared to send XML stanzas. [Section 4.2.2, RFC 6120] R: <stream: features/> NOTE: A <features/> element that contains only voluntary features indicates that the stream negotiation is complete and that the initiating entity is cleared to send XML stanzas. However, the initiating entity MAY negotiate further features if desired. [Section 4.2.2, RFC 6120]	2.6.3 IM-000100	R	R	L
1-11	On successful negotiation of a feature that necessitates a stream restart, both the initiating entity and the receiving entity SHALL consider the previous stream to be replaced, but SHALL NOT terminate the underlying TCP connection; instead, the initiating entity and the receiving entity SHALL reuse the existing connection. [Section 4.2.3, RFC 6120]	2.6.4 IM-000110	R	R	L
1-12	The initiating entity then SHALL send a new initial stream header to the receiving entity. [Section 4.2.3, RFC 6120]	2.6.4 IM-000120	R	R	L
1-13	When the receiving entity receives the new initial stream header, it SHALL generate a new stream ID (instead of reusing the old stream ID) and SHALL then send a new response stream header to the initiating entity. [Section 4.2.3, RFC 6120]	2.6.4 IM-000130	R	R	L
1-14	The receiving entity SHALL send an updated list of stream features to the initiating entity after a stream restart. [Section 4.2.4, RFC 6120]	2.6.5 IM-000140	R	R	L
1-15	The receiving entity SHALL indicate completion of the stream negotiation process by sending to the initiating entity either an empty <features/> element or a <features/> element that contains only voluntary features. Once stream negotiation is complete, the initiating entity is cleared to send XML stanzas over the stream for as long as the stream is maintained by both parties. [Section 4.2.5, RFC 6120] R: <stream: features/> NOTE: A <features/> element that contains only voluntary features indicates that the stream negotiation is complete and that the initiating entity is cleared to send XML stanzas, but that the initiating entity MAY negotiate further features if desired. [Section 4.2.5, RFC 6120]	2.6.5 IM-000150	R	R	L
1-16	For client-to-server sessions, a server SHALL allow a client to use "two streams over a single TCP connection." [Section 4.5, RFC 6120]	2.6.6 IM-000160	R		L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
1-17	For server-to-server sessions, the two server peers SHALL use two streams over two TCP connections, where one TCP connection is used for the stream in which stanzas are sent from the initiating entity to the receiving entity and the other TCP connection is used for the stream in which stanzas are sent from the receiving entity to the initiating entity. [Section 4.3, RFC 6120]	2.6.6 IM-000170	R	R	L
1-18	Client and server implementations SHALL be capable of closing an XML stream by sending a closing </stream> tag. [Section 4.4, RFC 6120] S: </stream:stream> NOTE: The entity that sends the closing stream tag SHOULD behave as follows [Section 4.4, RFC 6120]: a. Wait for the other party to close also its stream before terminating the underlying TCP connection (this gives the other party an opportunity to finish transmitting any data in the opposite direction before the TCP connection is terminated). b. Refrain from initiating the sending of further data over that stream but continue to process data sent by the other entity (and, if necessary, react to such data). c. Consider both streams to be void if the other party does not send its closing stream tag within a configurable amount of time. d. After receiving a reciprocal closing stream tag from the other party or waiting a configurable amount of time with no response, the entity SHALL terminate the underlying TCP connection.	2.6.7.1 IM-000180	R	R	L
1-19	After the entity that sent the first closing stream tag receives a reciprocal closing stream tag from the other party, it SHALL terminate the underlying TCP connection or connections. [Section 4.4, RFC 6120]	2.6.7.1 IM-000190	R	R	L
1-20	For client-to-server connections, it is assumed that the client knows the associated XMPP account name of the form <localpart@domain>. The client SHALL include the "from" attribute in the initial stream header it sends to the server and SHALL set the value to the associated XMPP account name of the form <localpart@domain>. [Section 4.6.1, RFC 6120]	2.6.8.1 IM-000200	R		L
1-21	For server-to-server connections, the initiating entity SHALL include the "from" attribute in the initial stream header it sends to the receiving entity and SHALL set its value to a hostname serviced by the initiating entity. [Section 4.6.1, RFC 6120]	2.6.8.1 IM-000210	R	R	L
1-22	For both client-to-server and server-to-server connections, the initiating entity SHALL include the "to" attribute in the initial stream header that it sends to the receiving entity and SHALL set its value to a hostname that the initiating entity knows or expects the receiving entity to service. [Section 4.6.2, RFC 6120]	2.6.8.1 IM-000220	R	R	L
1-23	For both client-to-server and server-to-server connections, the initiating entity SHALL include a "version" attribute whose value is "1.0" (or higher) in the initial stream headers it generates. [Section 4.6.5, RFC 6120] Example: C: <stream:stream from='john@im.example1.dod.mil' to='im.example1.dod.mil' version='1.0' xml:lang='en' xmlns='jabber: client' xmlns:stream='http://etherx.jabber.org/streams'>	2.6.8.1 IM-000230	R	R	L
1-24	For both client-to-server and server-to-server connections, the receiving entity SHALL include the "from" attribute in the response stream header that it sends to the initiating entity and SHALL set its value to a hostname serviced by the receiving entity. [Section 4.6.1, RFC 6120]	2.6.8.2 IM-000240	R	R	L
1-25	For response stream headers in client-to-server communication, if the client included a "from" attribute in the initial stream header then the server SHALL include a "to" attribute in the response stream header and SHALL set its value to the bare JID specified in the "from" attribute of the initial stream header. If the client did not include a "from" attribute in the initial stream header then the server SHALL NOT include a "to" attribute in the response stream header. [Section 4.6.2, RFC 6120]	2.6.8.2 IM-000250	R		L
1-26	For server-to-server connections, the receiving entity SHALL include the "to" attribute in the response stream header that it sends to the initiating entity and SHALL set its value to the hostname specified in the "from" attribute of the initial stream header. [Section 4.6.2, RFC 6120]	2.6.8.2 IM-000260	R	R	L
1-27	For both client-to-server and server-to-server connections, the receiving entity SHALL include an "id" attribute in the response stream header that it sends to the initiating entity. The "id" attribute communicates a unique identifier for the stream, called a STREAM ID. The stream "id" shall have the property of randomness. [Section 4.6.3, RFC 6120]	2.6.8.2 IM-000270	R	R	L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
1-28	For both client-to-server and server-to-server connections, the receiving entity SHALL include a "version" attribute where the value is 1.0 (or higher) in the response stream headers it sends to the initiating entity. [Section 4.6.5, RFC 6120] Example: S: <stream:stream from='im.example1.dod.mil' id='t7AMCin9zjMNwQKDnplntZPIDEI=' to='john@im.example1.dod.mil' version='1.0' xml:lang='en' xmlns='jabber: client' xmlns:stream='http://etherx.jabber.org/streams	2.6.8.2 IM-000280	R	R	L
1-29	Client and server implementations SHALL qualify the root <stream/> element ("stream header") by the namespace "http://etherx.jabber.org/streams" (the "streams namespace"). If this rule is violated, the entity that receives the offending stream header SHALL return a stream error to the sending entity, which SHALL be either <invalid-namespace/> or <bad-format/>. [Section 4.7.1, RFC 6120]	2.6.9.1 IM-000290	R		L
1-30	An entity (client or server) SHALL declare a content namespace for data sent over the stream. The content namespace SHALL be the same for the initial stream and the response stream so that both streams are qualified consistently. The content namespace applies to all first-level child elements sent over the stream unless explicitly qualified by another namespace. [Section 4.7.2, RFC 6120]	2.6.9.2 IM-000300	R	R	L
1-31	The XMPP defines two content namespaces: "jabber: client" and "jabber: server." Client implementations SHALL support the jabber: client content namespace. Server implementations SHALL support both the jabber: client content namespace (when the stream is used for communication between a client and a server) and the jabber:server content namespace (when the stream is used for communication between two servers). [Section 4.7.5, RFC 6120] Example: C: <stream:stream from='john@im.example1.dod.mil' to='im.example1.dod.mil' version='1.0' xml:lang='en' xmlns='jabber: client' xmlns:stream='http://etherx.jabber.org/streams'>	2.6.9.2 IM-000310	R	R	L
1-32	If an entity receives a first-level child element qualified by a content namespace it does not support, it SHALL return an <invalid-namespace/> stream error. [Section 4.7.5, RFC 6120]	2.6.9.2 IM-000320	R	R	L
1-33	The error child SHALL be sent by an entity (client or server) if it perceives that a stream-level error has occurred. [Section 4.8, RFC 6120]	2.6.10 IM-000330	R	R	L
1-34	Stream-level errors are unrecoverable. Therefore, if an error occurs at the level of the stream, the entity (client or server) that detects the error SHALL send an <error/> element with an appropriate child element that specifies the error condition and at the same time send a closing </stream> tag. [Section 4.8.1.1, RFC 6120] S: <stream:error> <xml-not-well-formed xmlns='urn:ietf:params:xml:ns:xmpp-streams'/> </stream:error> </stream:stream>	2.6.10 IM-000340	R	R	L
1-35	The entity that generates the stream error then SHALL close the stream as explained under Section 4.4 of RFC 6120). [Section 4.8.1.1, RFC 6120] C: </stream:stream>	2.6.10 IM-000350	R	R	L
1-36	If the error is triggered by the initial stream header, the receiving entity SHALL still send the opening <stream> tag, include the <error/> element as a child of the stream element, and then send the closing </stream> tag (preferably all at the same time). [Section 4.8.1.2, RFC 6120]	2.6.10 IM-000360	R	R	L
2	**2.7 - TLS and STARTTLS Negotiation**				
2-1	All XML streams (i.e., including both client-to-server and server-to-server connections) SHALL be secured with the use of the TLS protocol.	2.7 IM-000370	R	R	L
2-2	This specification mandates the use of the STARTTLS command to initiate TLS negotiation. All client and server implementations SHALL support and use the "STARTTLS" extension.	2.7.1 IM-000380	R	R	L
2-3	Immediately after the opening of the response stream, the receiving entity SHALL initiate the process of stream negotiation. [Section 5.4.1, RFC 6120]	2.7.1 IM-000390	R	R	L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
2-4	In the stream feature announcement provided by the receiving entity during the initial stage of the stream negotiation process, the receiving entity SHALL advertize ONLY the STARTTLS feature (qualified by the XML namespace: "urn:ietf:params:xml:ns:xmpp-tls") and SHALL also include an empty <required/> child element. [Section 5.4.1, RFC 6120] See the following example: R: <stream: features> <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'> <required/> </starttls> </stream: features>	2.7.1 IM-000400	R	R	L
2-5	In order to begin the STARTTLS negotiation, the initiating entity SHALL issue the STARTTLS command (i.e., a <starttls/> element qualified by the 'urn: ietf: params: xml: ns:xmpp-tls' namespace) to instruct the receiving entity that it wishes to begin a STARTTLS negotiation to secure the stream. [Section 5.4.2.1, RFC 6120] I: <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>	2.7.2 IM-000410	R	R	L
2-6	The receiving entity SHALL reply with a <proceed/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-tls' namespace. [Section 5.4.2.1, RFC 6120] R: <proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>	2.7.2 IM-000420	R	R	L
2-7	If there is a failure of STARTTLS negotiations, the receiving entity SHALL return a <failure/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-tls' namespace and SHALL close the XML stream. [Section 5.4.2.2, RFC 6120] R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/> R: </stream:stream>	2.7.3 IM-000430	R	R	L
2-8	After the receiving entity has sent and the initiating entity has received the <proceed/> element, the initiating and receiving entities SHALL proceed to TLS negotiation. The TLS negotiation and implementation SHALL be in accordance with all applicable DOD Security Technical Implementation Guideline (STIG) requirements [including DOD Public Key Infrastructure (PKI) compliance] and TLS/PKI implementation/ interoperability requirements as defined in Unified Capabilities Requirements (UCR) 2013, Section 4- Information Assurance.	2.7.4 IM-000440	R	R	L
2-9	If the TLS negotiation is successful, then the initiating and receiving entities SHALL proceed as follows. [Section 5.4.3.3, RFC 6120] The initiating entity SHALL send a new initial stream header to the receiving entity over the encrypted connection. The initiating entity SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the receiving entity and initiating entity MUST consider the original stream to be replaced upon success of the TLS negotiation. · The receiving entity SHALL respond with a new response stream header over the encrypted connection. In this new response stream header, the receiving entity SHALL generate a new stream ID instead of reusing the old stream ID. · The receiving entity also SHALL send stream features to the initiating entity, which SHALL NOT include the STARTTLS feature, but which SHALL advertise support of SASL negotiation as described in Section 2.8, Authentication and SASL Negotiation.	2.7.5 IM-000450	R	R	L
2-10	If the TLS negotiation results in failure, the receiving entity SHALL terminate the TCP connection. [Section 5.4.3.2, RFC 6120]	2.7.6 IM-000460	R	R	L
2-11	Client and server implementations SHALL complete STARTTLS negotiation before proceeding to SASL protocol negotiation; this order of negotiation is necessary to help safeguard authentication information sent during SASL negotiation, as well as to make it possible to base the use of the SASL EXTERNAL mechanism on a certificate provided during prior TLS negotiation (for entities who authenticate using a DOD PKI certificate). [Section 5.3.4, RFC 6120]	2.7.7 IM-000470	R		L
2-12	If the STARTTLS negotiation fails, the receiving entity SHALL return a <failure/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-tls' namespace, terminate the XML stream, and terminate the underlying TCP connection. [Section 5.4.2.2, RFC 6120]	2.7.8 IM-000480	R		L
3	**2.8 - Authentication and SASL Negotiation**				
3-1	All client and server implementations SHALL support SASL negotiations. [Section 6.2, RFC 6120]	2.8 IM-000490	R		L
3-2	The entities involved in an XML stream SHALL consider SASL as mandatory-to-negotiate. [Section 6.3.1, RFC 6120]	2.8 IM-000500	R	R	L
3-3	Anonymous login capability is prohibited. [Instant Messaging STIG, Version 1, Release 2]	2.8 IM-000510	R	R	L
3-4	During the prior TLS negotiation, the server SHALL authenticate using a DOD PKI certificate. The client SHALL validate the certificate presented by the server.	2.8.1 IM-000520	R		L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
3-5	The client SHALL authenticate using name and password using the SASL PLAIN mechanism [RFC 4616] as defined in the following text.	2.8.1 IM-000530	R		L
3-6	After successful STARTTLS negotiation, the server SHALL offer the SASL PLAIN mechanism to the client during SASL negotiation. The <mechanisms/> element SHALL be qualified by the 'urn: ietf: params: xml: ns: xmpp-sasl' namespace. The <mechanisms/> element SHALL contain one <mechanism/> child element including the appropriate value for the PLAIN mechanism. [Section 6.4.1, RFC 6120] S: <stream: features> <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'> <mechanism>PLAIN</mechanism> <required/> </mechanisms> </stream: features>	2.8.1 IM-000540	R		L
3-7	The client SHALL select the PLAIN authentication mechanism by sending an <auth/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace and which SHALL include the appropriate value for the PLAIN „mechanism" attribute. See the following example: C: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='PLAIN'>AGp1bGlldAByMG0zMG15cjBtMzA=</auth> As discussed in RFC 4616, the PLAIN SASL mechanism consists of a single message, a string of [UTF-8] encoded [Unicode] characters, from the client to the server. The client presents a NUL (U+0000) character, followed by the authentication identity (i.e., name), followed by a NUL (U+0000) character, followed by the clear-text password. For additional details, see RFC 4616. [Section 2, RFC 4616]	2.8.1 IM-000550	R		L
3-8	Upon receipt of the message, the server will verify the presented authentication identity and password by performing a directory lookup to a directory service linked to the XMPP server for authenticating the user. [Instant Messaging STIG, Version 1, Release 2]	2.8.1 IM-000560	R		L
3-9	All users SHALL be linked to a directory service, which is linked to the user's home XMPP server. [Instant Messaging STIG, Version 1, Release 2]	2.8.1 IM-000570	R		L
3-10	The server SHALL report the success of the handshake by sending a <success/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-sasl' namespace [Section 6.4.6. RFC 6120]: S: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>	2.8.1 IM-000580	R		L
3-11	After successful SASL negotiation, the client and server SHALL restart the stream. Upon receiving the <success/> element, the client SHALL initiate a new stream over the existing TLS connection by sending a new initial stream header to the server. The client SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the server and client MUST consider the original stream to be replaced upon sending or receiving the <success/> element. [Section 6.4.6, RFC 6120]	2.8.1 IM-000590	R		L
3-12	Upon receiving the new initial stream header from the client, the server SHALL respond by sending a new response stream header to the client (for which it SHALL generate a new stream ID instead of re-using the old stream ID). [Section 6.4.6, RFC 6120]	2.8.1 IM-000600	R		L
3-13	The server SHALL also send stream features, containing any further available features or containing no features (via an empty <features/> element). [Section 6.4.6, RFC 6120] S: <stream: features> <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/> </stream: features>	2.8.1 IM-000610	R	R	L
3-14	During the prior TLS negotiation, the initiating entity and the receiving entity SHALL mutually authenticate using DOD PKI certificates. Server-to-server mutual authentication SHALL be in accordance with all applicable DOD STIG requirements (including DOD PKI compliance) and TLS/PKI implementation/interoperability requirements as defined in UCR 2013, Section 4- Information Assurance.	2.8.2 IM-000620	R	R	L
3-15	After the successful mutual authentication of the receiving entity and the initiating entity during the prior TLS negotiation, the receiving entity SHALL offer the SASL EXTERNAL mechanism (as defined in Appendix A of RFC 4422) to the initiating entity during SASL negotiation. [Section 6.3.4, RFC 6120]	2.8.2 IM-000630	R	R	L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
3-16	The receiving entity SHALL include an empty <required/> element in its advertisement of the SASL feature. NOTE: The SASL EXTERNAL mechanism allows the initiating entity to request that the receiving entity use the credentials exchanged during the TLS Handshake process (See RFC 4422, Appendix A and XEP-0178: Best Practices for Use of SASL EXTERNAL With Certificates). R: <stream: features> <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'> <mechanism>EXTERNAL</mechanism> <required/> </mechanisms> </stream: features>	2.8.2 IM-000640	R	R	L
3-17	In response to the receiving entity offering the SASL EXTERNAL mechanism, the initiating entity SHALL select the EXTERNAL authentication mechanism by sending an <auth/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace and which SHALL include the appropriate value for the EXTERNAL „mechanism" attribute and which also includes an empty response of "=." [Section 6.4, RFC 6120 and Section 3, XEP-0178]: I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='EXTERNAL'/>=</auth> NOTE: For the sake of backwards compatibility, the initiating entity MAY alternatively include an authorization identity (base64-encoded as described in RFC 6120) as the XML character data of the <auth/> element, which SHOULD be the same as the „from" address in the stream header it sent to the initiating entity as defined in XEP-0178. I: <auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl' mechanism='EXTERNAL'>Y29uZmVyZW5jZS5leGFtcGxlLm9yZwo=</auth>	2.8.2 IM-000650	R	R	L
3-18	The receiving entity SHALL report the success of the handshake by sending a <success/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-sasl' namespace [Section 6.4.6, RFC 6120]: R: <success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>	2.8.2 IM-000660	R	R	L
3-19	After successful SASL negotiation, the initiating entity and the receiving entity SHALL restart the stream. Upon receiving the <success/> element, the initiating entity SHALL initiate a new stream over the existing TLS connection by sending a new initial stream header to the receiving entity. The initiating entity SHALL NOT send a closing </stream> tag before sending the new initial stream header, since the receiving entity and initiating entity MUST consider the original stream to be replaced upon sending or receiving the <success/> element. [Section 6.4.6, RFC 6120] I: <stream:stream from='im.example.dod.mil' to='chat.example2.dod.mil' version='1.0' xmlns='jabber:server' xmlns:stream='http://etherx.jabber.org/streams'>	2.8.2 IM-000670	R	R	L
3-20	Upon receiving the new initial stream header from the initiating entity, the receiving entity SHALL respond by sending a new response stream header to the initiating entity (for which it SHALL generate a new stream ID instead of reusing the old stream ID). [Section 6.3.2 and Section 6.4.6, RFC 6120] R: <stream from='im.example.dod.mil' id='MbbV2FeojySpUIP6J91qaa+TWHM=' to='chat.example2.dod.mil' version='1.0' xmlns='jabber:server' xmlns='http://etherx.jabber.org/streams'>	2.8.2 IM-000680	R	R	L
3-21	The receiving entity SHALL also send stream features, containing any further available features or containing no features (via an empty <features/> element). [Section 6.4.6, RFC 6120]	2.8.2 IM-000690	R	R	L
3-22	The receiving entity SHALL report failure of the handshake by sending a <failure/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-sasl' namespace. [Section 6.4.5, RFC 6120]	2.8.3 IM-000700	R	R	L
3-23	The particular cause of failure SHALL be communicated in an appropriate child element of the <failure/> element as defined under Section 6.4 (SASL Errors) of RFC 6120. [Section 6.4.5, RFC 6120] R: <failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'> not-authorized/> </failure>	2.8.3 IM-000710	R	R	L

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
3-24	The receiving entity SHALL allow a configurable number of retries (at least two and no more than three per IM STIG policy).	2.8.3 IM-000720	R	R	L
3-25	If the initiating entity exceeds the maximum number of retries, the server SHALL return a stream error (which SHALL be either <policy-violation/> or <not-authorized/>). [Section 6.4.5, RFC 6120]	2.8.3 IM-000730	R	R	L
4	**2.9 - Resource Binding**				
4-1	All client and server implementations SHALL support resource binding. [Section 7.2, RFC 6120]	2.9.2.1 IM-000740	R		L
4-2	For client-to-server connections, both the client and server SHALL consider resource binding as mandatory-to-negotiate. [Section 7.3.1, RFC 6120]	2.9.2.1 IM-000750	R		L
4-3	Upon sending a new response stream header to the client after successful SASL negotiation, the server SHALL include a <bind/> element qualified by the 'urn: ietf: params: xml: ns: xmpp-bind' namespace in the stream features it presents to the client. [Section 7.4, RFC 6120] S: <stream: features> <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/> </stream: features>	2.9.2.2 IM-000760	R		L
4-4	A server implementation SHALL be able to generate an XMPP resource part on behalf of a client. [Section 7.6, RFC 6120]	2.9.2.3 IM-000770	R		L
4-5	A resource part SHALL at a minimum, be unique among the connected resources for a specific local account in the form of <localpart@domain>. Enforcement of this policy is the responsibility of the server. [Section 7.5, RFC 6120]	2.9.2.3 IM-000780	R		L
4-6	A client SHALL request a server-generated resource part by sending an Info/Query (IQ) stanza of type "set" (see Section 2.11.2, Roster-Related Methods) containing an empty <bind/> element qualified by the 'urn:ietf:params:xml:ns:xmpp-bind' namespace. [Section 7.6.1, RFC 6120] C: <iq id='tn281v37' type='set'> <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/> </iq>	2.9.2.3 IM-000790	R		L
4-7	Once the server has generated an XMPP resource part for the client, it SHALL return an IQ stanza of type "result" to the client, which SHALL include a <jid/> child element that specifies the full JID for the connected resource as determined by the server. [Section 7.6.1, RFC 6120] S: <iq id='tn281v37' type='result'> <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'> <jid> juliet@im.example.com/4db06f06-1ea4-11dc-aca3-000bcd821bfb </jid> </bind> </iq>	2.9.2.3 IM-000800	R		L
5	**2.10 – XML Stanzas**				
5-1	Client and server implementations SHALL support the syntax and semantics associated with the message, presence, and IQ stanzas. [See the following Sections: 5.7.3.11.1 through 5.7.3.11.3] [Section 8, RFC 6120]	2.10 IM-000810	R	R	L
5-2	The following rules SHALL be followed regarding the use of the „to‟ attribute in the context of XML streams qualified by the „jabber: client‟ namespace (i.e., client-to-server streams) [Section 8.1.1.1, RFC 6120]: a. A stanza with a specific intended recipient SHALL possess a „to‟ attribute whose value is an XMPP address. b. A stanza sent from a client to a server for direct processing by the server on behalf of the client (e.g., presence sent to the server for broadcasting to other entities) SHALL NOT possess a „to‟ attribute.	2.10.1.1 IM-000820	R		T IO-4
5-3	The following rules SHALL be followed regarding the use of the „to‟ attribute in the context of XML streams qualified by the „jabber: server‟ namespace (i.e., server-to-server streams) [Section 8.1.1.2, RFC 6120]: a. A stanza SHALL possess a „to‟ attribute whose value is an XMPP address; if a server receives a stanza that does not meet this restriction, it SHALL generate an <improper-addressing/> stream error. b. The domain identifier portion of the JID in the „to‟ attribute SHALL match a hostname serviced by the receiving server; if a server receives a stanza that does not meet this restriction, it SHALL generate a <host-unknown/> or <host-gone/> stream error.	2.10.1.1 IM-000830	R	R	T IO-2, IO-4

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
5-4	The following rules SHALL be followed regarding the use of the „from‟ attribute in the context of XML streams qualified by the „jabber: client‟ namespace (i.e., client-to-server streams) [Section 8.1.2.1, RFC 6120]: a. When the server receives an XML stanza from a client, the server SHALL add a „from‟ attribute to the stanza or override the „from‟ attribute specified by the client, where the value of the „from‟ attribute is the full JID (<localpart@domainpart/resource>) determined by the server for the connected resource that generated the stanza or the bare JID (<localpart@domainpart>) in the case of subscription-related presence stanzas. b. When the server generates a stanza from the server itself for delivery to the client, the stanza SHALL include a „from‟ attribute whose value is the bare JID (i.e., <domain>) of the server as agreed upon during stream negotiation (e.g., based on the „to‟ attribute of the initial stream header). c. When the server generates a stanza from the server for delivery to the client on behalf of the account of the connected client (e.g., in the context of data storage services provided by the server on behalf of the client), the stanza SHALL either (a) not include a „from attribute or (b) include a 'from' attribute whose value is the account's bare JID (<localpart@domainpart>). d. A server SHALL NOT send to the client a stanza without a „from‟ attribute if the stanza was not generated by the server (e.g., if it was generated by another client or another server). e. When a client receives a stanza that does not include a „from‟ attribute, it SHALL assume that the stanza is from the user's account on the server.	2.10.1.2 IM-000840	R	R	T IO-4
5-5	The following rules SHALL be followed regarding the use of the „from‟ attribute in the context of XML streams qualified by the „jabber: server‟ namespace (i.e., server-to-server streams) [Section 8.1.2.2, RFC 6120]: a. A stanza SHALL possess a „from‟ attribute whose value is an XMPP address; if a server receives a stanza that does not meet this restriction, it SHALL generate an <improper-addressing/> stream error. b. The domain identifier portion of the JID contained in the „from‟ attribute SHALL match the hostname of the sending server (or any validated domain thereof) as communicated in the SASL negotiation; if a server receives a stanza that does not meet this restriction, it SHALL generate an <invalid-from/> stream error.	2.10.1.2 IM-000850	R	R	T IO-2, IO-4
5-6	For <iq/> stanzas, the originating entity SHALL include an „id‟ attribute. [Section 8.1.3, RFC 6120]	2.10.1.3 IM-000860	R	R	T IO-4
5-7	If the generated stanza includes an „id‟ attribute, then it is required for the associated response or error stanza to also include an „id‟ attribute, where the value of the „id‟ attribute SHALL match that of the generated stanza. [Section 8.1.3, RFC 6120]	2.10.1.3 IM-000870	R	R	T IO-4
5-8	If an inbound stanza received by a client or server does not possess an „xml:lang‟ attribute, an implementation SHALL assume that the default language is that which is specified for the stream. [Section 8.1.5, RFC 6120]	2.10.1.5 IM-000880	R	R	T IO-4
5-9	A server SHALL NOT modify or delete the „xml: lang‟ attribute of stanzas it receives from other entities. [Section 8.1.5, RFC 6120]	2.10.1.5 IM-000890	R	R	T IO-4
5-10	When a client or server implementation generates or processes an IQ stanza, the following rules apply [Section 8.2.3, RFC 6120]: a. An IQ stanza SHALL include the „id‟ attribute. b. An IQ stanza SHALL include the „type‟ attribute. c. The value of the „type‟ attribute for IQ stanzas SHALL be one of the following (if the value is other than one of the following strings, the recipient or an intermediate server SHALL return a stanza error of <bad-request/>): i. get – The stanza requests information (e.g., the stanza inquires about data which is needed in order to complete further operations) ii. set – The stanza provides data that is needed for an operation to be completed (e.g., it sets new values, replaces existing values) iii. result – The stanza is a response to a successful "get" or "set" request iv. error – The stanza reports an error that has occurred regarding the processing or delivery of a previously sent "get" or "set" request d. An entity that receives an IQ request of type "get" or "set" SHALL reply with an IQ response of type "result" or "error." The response SHALL preserve the 'id' attribute of the request. e. An entity that receives a stanza of type "result" or "error" SHALL NOT respond to the stanza by sending a further IQ response of type "result" or "error." f. An IQ stanza of type "gets" or "set" SHALL contain exactly one child element, which specifies the semantics of the particular request. g. An IQ stanza of type "result" SHALL include zero or one child element. h. An IQ stanza of type "error" SHALL include an <error/> child.	2.10.2.3 IM-000900	R	R	T IO-4

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
5-11	Client and server implementations SHALL comply with the mandatory requirements defined in Section 8.3 of RFC 6120.	2.10.3 IM-000910	R		T IO-4
5-12	If the domain part of the JID contained in the „to‟ attribute does not match one of the configured hostnames of the server itself, the server SHALL attempt to route the stanza to the remote domain. [Section 10.4, RFC 6120]	2.10.4.1 IM-000920	R	R	T IO-4
5-13	If a server-to-server stream already exists between the two domains, the sender's server SHALL attempt to route the stanza to the authoritative server for the remote domain over the existing stream. [Section 10.4.1, RFC 6120]	2.10.4.1.1 IM-000930	R	R	T IO-2, IO-4
5-14	If no server-to-server stream exists between the two domains, the sender's server SHALL proceed as follows [Section 10.4.2, RFC 6120]: · Resolve the hostname of the remote domain, as described in Section 2.6.11, Hostname Resolution. · Negotiate a server-to-server stream between the two domains (as defined in Section 2.7, TLS and STARTTLS Negotiation, and Section 2.8, Authentication and SASL Negotiation. · Route the stanza to the authoritative server for the remote domain over the newly-established stream.	2.10.4.1.2 IM-000940	R	R	T IO-2, IO-4
5-15	If the routing of a stanza to the intended recipient's server is unsuccessful, the sender's server SHALL return an error to the sender. If resolution of the remote domain is unsuccessful, the stanza error SHALL be <remote-server-not-found/>. If the resolution succeeds, but the XML streams cannot be negotiated, the stanza error SHALL be <remote-server-timeout/>. [Section 10.4.3, RFC 6120]	2.10.4.1.3 IM-000950	R	R	T O-2, IO-4
5-16	If stream negotiation with the intended recipient's server is successful but the remote server cannot deliver the stanza to the recipient, the remote server SHALL return an appropriate error to the sender by way of the sender's server. [Section 10.4.3, RFC 6120]	2.10.4.1.3 IM-000960	R	R	T IO-2, IO-4
5-17	If the hostname of the domain part of the JID contained in the „to‟ attribute matches one of the configured hostnames of the server, the server SHALL first determine if the hostname is serviced by the server itself or by a specialized local service. If the latter, the server SHALL route the stanza to that service. If the former, the server SHALL proceed as follows [Section 10.5.3, RFC 6120]:	2.10.4.2 IM-000970	R	R	T IO-4
5-18	If there is no local account associated with the <localpart@domainpart>, how the stanza is processed depends on the stanza type. [Section 10.5.3.1, RFC 6120] · For a message stanza, the server SHALL return a <service-unavailable/> stanza error to the sender. · For a presence stanza, the server SHALL ignore the stanza. · For an IQ stanza, the server SHALL return a <service-unavailable/> stanza error to the sender.	2.10.4.2.1 IM-000980	R	R	T IO-4
5-19	If the JID contained in the „to‟ attribute is of the form <localpart@domainpart>, how the stanza is processed depends on the stanza type. [Section 10.5.3.2, RFC 6120] · For a message stanza, if at least one connected resource for the account exists, the server SHALL deliver it to at least one of the connected resources. If there exists no connected resource, the server SHALL either return a <service-unavailable/> stanza error or store the message offline for delivery when the account next has a connected resource. · For a presence stanza, if at least one connected resource that has sent initial presence exists (i.e., has a "presence session"), the server SHALL deliver it to such resources. If no connected resource exists, the server SHALL ignore the stanza. For an IQ stanza, the server SHALL handle it directly on behalf of the intended recipient.	2.10.4.2.2 IM-000990	R	R	T IO-4
5-20	If the JID contained in the „to‟ attribute is of the form <localpart@domainpart/resource> and there is no connected resource that exactly matches the full JID, the stanza SHALL be processed as if the JID were of the form <localpart@domainpart>. [Section 10.5.4, RFC 6120]	2.10.4.2.3 IM-001000	R	R	T IO-4
5-21	If the JID contained in the „to‟ attribute is of the form <localpart@domainpart/resource> and there is a connected resource that exactly matches the full JID, the server SHALL deliver the stanza to that connected resource. [Section 10.5.4, RFC 6120]	2.10.4.2.3 IM-001010	R	R	T IO-4
6	**2.11 – Roster Management**				
6-1	Client and server implementations SHALL use IQ stanzas containing a <query/> child element qualified by the „jabber: iq: roster‟ namespace to manage elements in a roster. [Section 2.1, RFC 6121]	2.11.1 IM-001020	R		T IO-6

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
6-2	Client and server implementations SHALL support the „subscription‟ attribute and the allowable subscription-related values for this attribute. The state of the presence subscription in relation to a roster item is captured in the „subscription‟ attribute of the <item/> element. The allowable subscription-related values for this attribute are [Section 2.1.2.5, RFC 6121]: a. "none" – the user does not have a subscription to the contact's presence, and the contact does not have a subscription to the user's presence; this is the default value, so if the subscription attribute is not included, then the state is to be understood as "none." b. "to" – the user has a subscription to the contact's presence, but the contact does not have a subscription to the user's presence. c. "from – the contact has a subscription to the user's presence, but the user does not have a subscription to the contact's presence. d. "both" – both the user and the contact have subscriptions to each other's presence (also called a "mutual subscription").	2.11.1 IM-001030	R		T IO-6
6-3	In a roster result, the client SHALL ignore values of the „subscription‟ attribute other than "none", "to", "from", or "both." [Section 2.1.2.5, RFC 6121]	2.11.1 IM-001040	R		T IO-6
6-4	In a roster push, the client SHALL ignore values of the „subscription‟ attribute other than "none", "to", "from", "both", or "remove." [Section 2.1.2.5, RFC 6121]	2.11.1 IM-001050	R		T IO-6
6-5	In a roster set, the value of the „subscription‟ can have a value of "remove", which indicates that the item is to be removed from the roster; the server SHALL ignore all values of the „subscription‟ attribute other than "remove." [Section 2.1.2.5, RFC 6121]	2.11.1 IM-001060	R		T IO-6
6-6	Client implementations SHALL support the „name‟ attribute, which is used to specify the "handle" to be associated with the JID, as determined by the user (not the contact). It is optional for a client to include the „name‟ attribute when adding or updating a roster item. [Section 2.1.2.4, RFC 6121]	2.11.1 IM-001070	R		T IO-6
6-7	Client and server implementations SHALL support the „ask‟ attribute, which is used to specify presence subscriptions sub-state. [Section 2.1.2.2, RFC 6121]	2.11.1 IM-001080	R		T IO-6
6-8	A value of "subscribe" in the „ask‟ attribute is used to signal a "Pending Out" sub-state as described under Section 3.1.2 of RFC 6121. A server SHALL include the „ask‟ attribute to inform the client of "Pending Out" sub-state. [Section 2.1.2.2, RFC 6121]	2.11.1 IM-001090	R		T IO-6
6-9	Client and server implementations SHALL support the <group/> child element which is used to specify a category or "bucket" into which the roster item is to be grouped by a client. It is optional for a client to include the <group/> element when adding or updating a roster item. If a roster set (Roster Set) includes no <group/> element, then the item is to be interpreted as being affiliated with no group. [Section 2.1.2.6, RFC 6121]	2.11.1 IM-001100	R		T IO-6
6-10	A client implementation SHALL have the ability to generate a Roster Get. A Roster Get is a client's request for the server to return the roster; syntactically it is an IQ stanza of type "get" sent from client to server and containing a <query/> element qualified by the „jabber: iq: roster‟ namespace, where the <query/> element SHALL NOT contain any <item/> child elements. Likewise, a compliant server implementation SHALL be able to process this request. The expected outcome of sending a roster get is for the server to return a roster result. [Section 2.1.3, RFC 6121] C: <iq from='john.smith@chat.dod.mil/desktop client' id='bv1bs71f' type='get'> <query xmlns='jabber:iq:roster'/> </iq>	2.11.2 IM-001110	R		T IO-6
6-11	A server implementation SHALL be able to process a Roster Get.	2.11.2 IM-001120	R		L/ T IO-6

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
6-12	A server implementation SHALL have the ability to generate a Roster Result. A Roster Result is the server's response to a roster get; syntactically it is an IQ stanza of type "result" sent from server to client and containing a <query/> element qualified by the „jabber:iq:roster‟ namespace. The <query/> element in a roster result contains one <item/> element for each contact and therefore can contain more than one <item/> element. The ability to generate this response is required for server implementations. Likewise, a compliant client implementation SHALL be able to process this response. [Section 2.1.4, RFC 6121] S: <iq id='bv1bs71f' to='robert.jones@chat.dod.mil/desktop client‟ type='result'> <query xmlns='jabber:iq:roster' ver='ver7'> <item jid='mike@example2.dod.mil'/> <item jid='bob@example1.dod.mil'/> </query> </iq>	2.11.2 IM-001130	R		T IO-6
6-13	A client implementation SHALL be able to process a Roster Result.	2.11.2 IM-001140	R		T IO-6
6-14	A client implementation SHALL have the ability to generate a Roster Set. A Roster Set is a client's request for the server to modify (i.e., create, update, or delete) a roster item; syntactically it is an IQ stanza of type "set" sent from client to server and containing a <query/> element qualified by the „jabber:iq:roster‟ namespace. [Section 2.1.5, RFC 6121] The following rules apply to roster sets: a. The <query/> element SHALL contain one and only one <item/> element. b. The server SHALL ignore any value of the „subscription‟ attribute other than "remove." C: <iq from='robert@example2.dod.mil' id='rs1' type='set'> <query xmlns='jabber:iq:roster'> <item jid='bob@chat.dod.mil'/> </query> </iq>	2.11.2 IM-001150	R		T IO-6
6-15	A server implementation SHALL be able to process a Roster Set.	2.11.2 IM-001160	R		T IO-6
6-16	A server implementation SHALL have the ability to generate a Roster Push. A Roster Push is a newly created, updated, or deleted roster item that is sent from the server to the client; syntactically it is an IQ stanza of type "set" sent from server to client and containing a <query/> element qualified by the „jabber:iq:roster‟ namespace. [Section 2.1.6, RFC 6121] The following rules apply to roster pushes: a. The <query/> element in a roster push SHALL contain one and only one <item/> element. b. A receiving client SHALL ignore the stanza unless it has no „from‟ attribute (i.e., implicitly from the user's bare JID) or it has a „from‟ attribute whose value matches the user's bare JID <user@domain>. S: <iq id='a78b4q6ha463' to='john@example1.dod.mil/desktop client' type='set'> <query xmlns='jabber:iq:roster'> <item jid='robert@example2.dod.mil'/> </query> </iq>	2.11.2 IM-001170	R		T IO-6
6-17	A client implementation SHALL be able to process a Roster Push.	2.11.2 IM-001180	R		T IO-6
6-18	As mandated by the semantics of the IQ stanza as defined in [RFC 6120] each resource that receives a roster push SHALL reply with an IQ stanza of type „result‟ (or „error‟). C: <iq from='john@example1.dod.mil/desktop client' id='a78b4q6ha463' type='result'/>	2.11.2 IM-001190	R		T IO-6
6-19	Upon authenticating with a server and binding a resource (thus becoming a connected resource), a client SHALL request the roster before sending initial presence. A client requests the roster by sending a roster get over its stream to the server. [Section 2.2, RFC 6121]	2.11.3 IM-001200	R		T IO-6

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
6-20	The server SHALL process the roster get and SHALL return a roster result containing a <query/> element qualified by the „jabber:iq:roster‟ namespace. The <query/> element in a roster result SHALL contain one <item/> element for each contact and therefore can contain more than one <item/> element. [Section 2.1.3 and Section 2.2, RFC 6121] C: <iq from='john@example1.dod.mil' id='hu2bac18' type='get'> <query xmlns='jabber:iq:roster'/> </iq> S: <iq id='hu2bac18' to='john@example1.dod.mil/desktop client' type='result'> <query xmlns='jabber:iq:roster' ver='ver11'> <item jid='robert@example2.dod.mil' name='Robert' subscription='both'> <group>Friends</group> </item> <item jid='mike@example2.dod.mil' name='Mike' subscription='from'/> <item jid='bob@example1.dod.mil' name='Bob' subscription='both'/> </query> </iq>	2.11.3 IM-001210	R		T IO-6
6-21	If the server cannot process the roster get, it SHALL return an appropriate stanza error as described in RFC 6120.	2.11.3 IM-001220	R		T IO-6
6-22	A client SHALL support the ability to add an item to the roster by sending a roster set containing a new item. [Section 2.3.1, RFC 6121] C: <iq from='john@example1.dod.mil/desktop client' id='ph1xaz53' type='set'> <query xmlns='jabber:iq:roster'> <item jid='robert@example2.dod.mil' name='Robert'> <group>Friends</group> </item> </query> </iq>	2.11.4 IM-001230	R		T IO-6
6-23	If the server can successfully process the roster set for the new item (i.e., if no error occurs), it SHALL create the roster item in persistent storage. The server SHALL then return an IQ stanza of type "result" to the connected resource that sent the roster set. [Section 2.3.2, RFC 6121]	2.11.4 IM-001240	R		T IO-6
6-24	The server SHALL also send a roster push containing the new roster item to all of the user's interested resources, including the resource that generated the roster set. [Section 2.3.2, RFC 6121]	2.11.4 IM-001250	R		T IO-6
6-25	If the server cannot successfully process the roster set, it SHALL return a stanza error. For additional details, see Section 2.3.3 of RFC 6121.	2.11.4 IM-001260	R		T IO-6
6-26	A client SHALL support the ability to update a roster item by sending a roster set to the server. Because a roster item is atomic, the item SHALL be updated exactly as provided in the roster set. [Section 2.4.1, RFC 6121] a. Adding a group. b. Deleting a group.	2.11.5 IM-001270	R		T IO-6
6-27	As with adding a roster item, if the roster item can be successfully processed, then the server SHALL update the roster information in persistent storage, send a roster push to the entire user's interested resources, and send an IQ result to the initiating resource. [Section 2.4.2, RFC 6121]	2.11.5 IM-001280	R		T IO-6

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
6-28	A client SHALL support the ability to delete a roster item by sending a roster set and specifying the value of the „subscription‟ attribute to "remove." [Section 2.5.1, RFC 6121] C: <iq from='john@example1.dod.mil/desktop client' id='hm4hs97y' type='set'> <query xmlns='jabber:iq:roster'> <item jid='robert@example2.dod.mil' subscription='remove'/> </query> </iq>	2.11.6 IM-001290	R		T IO-6
6-29	As with adding a roster item, if the server can successfully process the roster set then it SHALL update the roster information in persistent storage, send a roster push to all of the user's interested resources (with the „subscription‟ attribute set to a value of „remove‟), and send an IQ result to the initiating resource. [Section 2.5.2, RFC 6121]	2.11.6 IM-001300	R		T IO-6
6-30	The user's server SHALL generate one or more subscription-related presence stanzas, as per the following use cases [Section 2.5.2, RFC 6121]: a. If the user has a presence subscription to the contact, then the user's server SHALL send a presence stanza of type "unsubscribe" to the contact (to unsubscribe from the contact's presence). b. If the contact has a presence subscription to the user, then the user's server SHALL send a presence stanza of type "unsubscribed" to the contact (to cancel the contact's subscription to the user), or both. c. If the presence subscription is mutual, then the user's server SHALL send both a presence stanza of type "unsubscribe" and a presence stanza of type "unsubscribed" to the contact. S: <presence from='john@example1.dod.mil' id='lm3ba81g' to='robert@example2.dod.mil' type='unsubscribe'/>	2.11.6 IM-001310	R		T IO-6
6-31	If the value of the „jid‟ attribute specifies an item that is not in the roster, then the server SHALL return an <item-not-found/> stanza error. [Section 2.5.3, RFC 6121]	2.11.6 IM-001320	R		T IO-6
7	**2.12 – Presence Subscription Management**				
7-1	A client implementation SHALL be capable of generating a subscription request by sending a presence stanza of type "subscribe." [Section 3.1.1, RFC 6121] UC: <presence id='xk3h1v69' to='john@example1.dod.mil' type='subscribe'/>	2.12.1.1 IM-001330	R		T IO-4, IO-3
7-2	When the client sends a presence subscription request to a potential instant messaging and presence contact, the value of the „to‟ attribute SHALL be a bare JID <contact@domain> rather a full JID <contact@domain/resource>. [Section 3.1.1, RFC 6121]	2.12.1.1 IM-001340	R		T IO-4, IO-3
7-3	Upon receiving the outbound presence subscription request, the user's server SHALL comply with the following rules for Server Processing of Outbound Subscription Requests as defined below [Section 3.1.2, RFC 6121]: a. Before processing the request, the user's server SHALL check the syntax of the JID contained in the „to‟ attribute. If the JID is of the form <localpart@domain/resource part> instead of <localpart@domain>, the user's server SHALL treat it as if the request had been directed to the contact's bare JID and modify the „to‟ address accordingly. b. If the potential contact is hosted on the same server as the user, then the server SHALL adhere to the Rules for Server Processing of Inbound Subscription Requests (see below) and SHALL deliver it to the local contact. c. If the potential contact is hosted on a remote server, the user's server SHALL then route the stanza to that remote domain in accordance with the Server Rules for Processing XML Stanzas (e.g., see Section 2.10.4.1, Rules for Processing XML Stanzas to Remote Domains).	2.12.1.2 IM-001350	R		T IO-4, IO-3
7-4	When a server processes or generates an outbound presence stanza of type "subscribe", "subscribed", "unsubscribe", or "unsubscribed", the server SHALL stamp the outgoing presence stanza with the bare JID <localpart@domain> of the sending entity. Enforcement of this rule simplifies the presence subscription model and helps to prevent presence leaks. [Section 3.1.2, RFC 6121]	2.12.1.2 IM-001360	R		T IO-4, IO-3
7-5	If the presence subscription request cannot be locally delivered or remotely routed (e.g., because the request is malformed, the local contact does not exist, the remote server does not exist, an attempt to contact the remote server times out, or any other error determined or experienced by the user's server), then the user's server SHALL return an appropriate error stanza to the user. [Section 3.1.2, RFC 6121]	2.12.1.2 IM-001370	R		T IO-2, IO-4, IO-3

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
7-6	After locally delivering or remotely routing the presence subscription request, the user's server SHALL then send a roster push to all of the user's interested resources, containing the potential contact with a subscription state of "none" and with notation that the subscription is pending (via an „ask‟ attribute whose value is "subscribe"). [Section 3.1.2, RFC 6121]: US: <iq id='b89c5r7ib574' to='john.smith@chat.dod.mil/desktop client' type='set'> <query xmlns='jabber:iq:roster'> <item ask='subscribe' jid=„robert.jones@example2.dod.mil/desktop client‟ subscription='none'/> </query> </iq>	2.12.1.2 IM-001380	R		T IO-4, IO-3
7-7	Before processing the inbound presence subscription request, the contact's server SHALL check the syntax of the JID contained in the „to‟ attribute. If the JID is of the form <contact@domain/resource> instead of <contact@domain>, the contact's server SHALL treat it as if the request had been directed to the contact's bare JID and modify the „to‟ address accordingly. [Section 3.1.3, RFC 6121]	2.12.1.3 IM-001390	R		T IO-4, IO-3
7-8	When processing the inbound presence subscription request, the user's server SHALL comply with the following rules for Server Processing of Inbound Subscription Requests as defined below [Section 3.1.3, RFC 6121]: a. Above all, the contact's server SHALL NOT automatically approve subscription requests on the contact's behalf (unless the contact has configured its account to automatically approve subscription requests). Instead, the contact's server SHALL deliver that request to the contact's available resource(s) for approval or denial by the contact. b. If the contact exists and the user already has a subscription to the contact's presence, then the contact's server SHALL auto-reply on behalf of the contact by sending a presence stanza of type "subscribed" from the contact's bare JID to the user's bare JID. c. If the contact does not exist, then the contact's server SHALL automatically return a presence stanza of type "unsubscribed" to the user. d. Otherwise, if there is at least one available resource associated with the contact when the subscription request is received by the contact's server, then the contact's server SHALL broadcast that subscription request to all of the contact's available resources. e. Otherwise, if the contact exists, the user does not already have a subscription to the contact's presence, and the contact has no available resources when the subscription request is received by the contact's server, then the contact's server SHALL keep a record of the complete presence stanza comprising the subscription request, including any extended content contained therein, and deliver the request when the contact next has an available resource. The contact's server SHALL continue to deliver the subscription request whenever the contact creates an available resource, until the contact either approves or denies the request.	2.12.1.3 IM-001400	R		T IO-4, IO-3
7-9	When the contact's client receives a subscription request from the user, it SHALL present the request to the contact for approval (unless the contact has explicitly configured the client to automatically approve or deny some or all subscription requests). [Section 3.1.4, RFC 6121]	2.12.1.4 IM-001410	R		T IO-4, IO-3
7-10	A client implementation SHALL be capable of generating a subscription approval by sending a presence stanza of type "subscribed." CC: <presence id='h4v1c4kj' to='robert@example2.dod.mil' type='subscribed'/>	2.12.1.4 IM-001420	R		T IO-4, IO-3
7-11	A client implementation SHALL be capable of denying a subscription request by sending a presence stanza of type "unsubscribed." [Section 3.1.4, RFC 6121] CC: <presence id='h4v1c4kj' to='robert@example2.dod.mil' type='unsubscribed'/>	2.12.1.4 IM-001430	R		T IO-4, IO-3
7-12	When the contact's client sends the subscription approval, the contact's server SHALL stamp the outbound stanza with the bare JID <localpart@domain> of the contact and locally deliver or remotely route the stanza to the user. [Section 3.1.5, RFC 6121] CS: <presence from='john@example1.dod.mil' id='h4v1c4kj' to='robert@example2.dod.mil' type='subscribed'/>	2.12.1.5 IM-001440	R		T IO-4, IO-3

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
7-13	The contact's server then SHALL send an updated roster push to all of the contact's interested resources, with the „subscription‟ attribute set to a value of "from." [Section 3.1.5, RFC 6121]	2.12.1.5 IM-001450	R		T IO-4, IO-3
7-14	The contact's server SHALL then also send current presence to the user from each of the contact's available resources. [Section 3.1.5, RFC 6121]	2.12.1.5 IM-001460	R		T IO-4, IO-3
7-15	When the user's server receives the subscription approval, it SHALL first check if the contact is in the user's roster with subscription=„none‟ or subscription=„from‟ and the „ask‟ flag set to "subscribe" (see Appendix A of RFC 6121). If this check is successful, then the user's server SHALL proceed as follows [Section 3.1.6, RFC 6121]: a. Deliver the inbound subscription approval to all of the user's interested resources. This SHALL occur before sending the roster push described in the next step. [Section 3.1.6, RFC 6121] US: <presence from='john@example1.dod.mil' id='h4v1c4kj' to='robert@example2.dod.mil' type='subscribed'/> b. Initiate a roster push to all of the user's interested resources, containing an updated roster item for the contact with the „subscription‟ attribute set to a value of "to" (if the subscription state was "None + Pending Out" or "None + Pending Out + In") or "both" (if the subscription state was "From + Pending Out"). See Table 5 of Appendix A of RFC 6121. [Section 3.1.6, RFC 6121] US: <iq id='b89c5r7ib576' to='robert@example2.dod.mil/desktop client' type='set'> <query xmlns='jabber: iq: roster'> <item jid='john@example1.dod.mil' subscription='to'/> </query> </iq> The user's server SHALL also deliver the available presence stanza received from each of the contact's available resources to each of the user's available resources.	2.12.1.6 IM-001470	R		T IO-4, IO-3
7-16	Otherwise – that is, if the user does not exist, if the contact is not in the user's roster, or if the contact is in the user's roster with a subscription state other than those described in the foregoing check – then the user's server SHALL silently ignore the subscription approval stanza by not delivering it to the user, not modifying the user's roster, and not generating a roster push to the user's interested resources. [Section 3.1.6, RFC 6121]	2.12.1.6 IM-001480	R		T IO-4, IO-3
7-17	A client implementation SHALL be capable of sending a presence stanza of type "unsubscribed" in order to cancel a subscription that it has previously granted to a user. [Section 3.2.1, RFC 6121] CC: <presence id='ij5b1v7g' to='robert@example2.dod.mil' type='unsubscribed'/>	2.12.2.1 IM-001490	R		T IO-4, IO-3
7-18	Upon receiving the outbound subscription cancellation, the contact's server SHALL proceed as follows [Section 3.2.2, RFC 6121]: a. If the user is hosted on the same server as the contact, then the server SHALL adhere to the rules specified in the next section in processing the subscription cancellation. b. If the user is hosted on a remote server, the contact's server SHALL then route the stanza to that remote domain. c. As mentioned, before locally delivering or remotely routing the stanza, the contact's server SHALL stamp the outbound subscription cancellation with the bare JID <localpart@domain> of the contact. CS: <presence from='john@example1.dod.mil' id='ij5b1v7g' to='robert@example2.dod.mil' type='unsubscribed'/> d. The contact's server then SHALL send a roster push with the updated roster item to all of the contact's interested resources, where the subscription state is now either "none" or "to." For added clarification, see Appendix A of RFC 6121. e. The contact's server then SHALL send a presence stanza of type "unavailable" from all of the contact's online resources to the user. CS: <presence from='john@example1.dod.mil/desktop client' id='i8bsg3h3' type='unavailable'/>	2.12.2.2 IM-001500	R		T IO-4, IO-3

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
7-19	When the user's server receives the inbound subscription cancellation, it SHALL first check if the contact is in the user's roster with subscription=„to‟ or subscription=„both‟ (see Appendix A of RFC 6121).	2.12.2.3 IM-001510	R		T IO-4, IO-3
7-20	To unsubscribe from a contact's presence, the client SHALL send a presence stanza of type "unsubscribe." [Section 3.3.1, RFC 6121] UC: <presence id='ul4bs71n' to='john@example.dod.mil' type='unsubscribe'/>	2.12.3.1 IM-001520	R		T IO-4, IO-3
7-21	Upon receiving the outbound unsubscribe, the user's server SHALL proceed as follows [Section 3.3.2, RFC 6121]: a. If the contact is hosted on the same server as the user, then the server SHALL adhere to the rules specified for Server Processing of Inbound Unsubscribe (see below). b. If the contact is hosted on a remote server, the user's server SHALL then route the stanza to that remote domain. c. The user's server then SHALL send a roster push with the updated roster item to all the user's interested resources, where the subscription state is now either "none" or "from" (see Appendix A of RFC 6121). US: <iq id='h37h3u1bv402' to='robert@example2.dod.mil/desktop client' type='set'> <query xmlns='jabber:iq:roster'> <item jid='john@example1.dod.mil' subscription='none'/> </query> </iq>	2.12.3.2 IM-001530	R		T IO-4, IO-3
7-22	When the contact's server receives the unsubscribe notification, it SHALL first check if the user is in the contact's roster with subscription=„from‟ or subscription=„both‟ (i.e., a subscription state of "From", "From + Pending Out", or "Both"; see Appendix A of RFC 6121). a. If this check is successful, the contact's server SHALL [Section 3.3.3, RFC 6121]: i. Deliver the inbound unsubscribe to all of the contact's interested resources. This SHALL occur before sending the roster push described in the next step. ii. Initiate a roster push to all of the contact's interested resources, containing an updated roster item for the contact with the „subscription‟ attribute set to a value of "none" (if the subscription state was "From" or "From + Pending Out") or "to" (if the subscription state was "Both"). iii. Generate an outbound presence stanza of type "unavailable" from each of the contact's available resources to the user. b. If the check (above) is not successful, that is, if the contact does not exist, if the user is not in the contact's roster, or if the user is in the contact's roster with a subscription state other than those described in the foregoing check, then the contact's server SHALL silently ignore the stanza by not delivering it to the contact, not modifying the contact's roster, and not generating a roster push to the contact's interested resources. [Section 3.3.3, RFC 6121]	2.12.3.3 IM-001540	R		T IO-4, IO-3
8	**2.13 – Exchanging Presence Information**				
8-1	After completing the mandatory-to-negotiate stream features and retrieving a roster, a client implementation SHALL signal its availability for communication by sending initial presence to its server, i.e., a presence stanza with no „to‟ address and no „type‟ attribute. [Section 4.2.1, RFC 6121] UC: <presence/> NOTE: The initial presence stanza may contain the <priority/> element, the <show/> element, and one or more instances of the <status/> element. [Section 4.2, RFC 6121]	2.13.1.1 IM-001550	R		T IO-4, IO-3
8-2	Upon receiving initial presence from a client, the user's server SHALL send the initial presence stanza from the full JID <user@domain/resource> of the user to all contacts that are subscribed to the user's presence. [Section 4.2.2, RFC 6121] US: <presence from='user@domain/resource part' to='contact@domain'/>	2.13.1.2 IM-001560	R		T IO-4, IO-3
8-3	The user's server SHALL also broadcast initial presence from the user's newly available resource to all of the user's available resources (including the resource that generated the presence notification in the first place). [Section 4.2.2, RFC 6121]	2.13.1.2 IM-001570	R		T IO-4, IO-3
8-4	In the absence of presence information about the user's contacts, the user's server SHALL also send presence probes to the user's contacts on behalf of the user (see Section 2.13.2, Presence Probes). [Section 4.2.2, RFC 6121]	2.13.1.2 IM-001580	R		T IO-4, IO-3
8-5	Upon receiving presence from the user, the contact's server SHALL deliver the user's presence stanza to all of the contact's available resources. [Section 4.2.3, RFC 6121]	2.13.1.3 IM-001590	R		T IO-4, IO-3

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
8-6	When the contact's client receives presence from the user, it SHALL proceed as follows [Section 4.2.4, RFC 6121]: a. If the user is in the contact's roster, the client SHALL display the presence information in an appropriate roster interface. b. If the user is not in the contact's roster, the client SHALL ignore the presence information and not display it to the contact.	2.13.1.4 IM-001600	R		T IO-4, IO-3
8-7	To discover the availability of a user's contact, the user's server SHALL be capable of sending a presence probe from the bare JID <user@domain> of the user to the bare JID <contact@domain> of the contact. [Section 4.3.1, RFC 6121] US: <presence from='john@example1.dod.mil' id='ign291v5' to='robert@example2.dod.mil' type='probe'/>	2.13.2.1 IM-001610	R		T IO-4, IO-3
8-8	The server SHALL NOT send a probe to a contact if the user is not subscribed to the contact's presence (i.e., if the contact is not in the user's roster with the „subscription" attribute set to a value of "to" or "both"). [Section 4.3.1, RFC 6121]	2.13.2.1 IM-001620	R		T IO-4, IO-3
8-9	Upon receiving a presence probe to the contact's bare JID from the user's server on behalf of the user, the contact's server SHALL reply as follows [Section 4.3.2, RFC 6121]: a. If the contact account does not exist or the user is in the contact's roster with a subscription state other than "From", "From + Pending Out", or "Both" (as defined under Appendix A of RFC 6121), then the contact's server SHALL return a presence stanza of type "unsubscribed" in response to the presence probe. Here the „from" address SHALL be the bare JID of the contact, since specifying a full JID would constitute a presence leak as described in RFC 6120. CS: <presence from='mike@example2.dod.mil' id='xv291f38' to='john@example1.dod.mil' type='unsubscribed'/>2. b. Else, if the contact has no available resources, then the server SHALL reply to the presence probe by sending to the user a presence stanza of type "unavailable." c. Else, if the contact has at least one available resource, then the server SHALL reply to the presence probe by sending to the user the full XML of the last presence stanza with no „to" attribute received by the server from each of the contact's available resources. Here the „from" addresses are the full JIDs of each available resource. CS: <presence from='robert@example2.dod.mil/foo' id='hzf1v27k' to='john@example1.dod.mil'/>	2.13.2.2 IM-001630	R		T IO-4, IO-3
8-10	After sending initial presence, a client implementation SHALL be capable of updating its availability by sending a presence stanza with no „to" address and no „type" attribute. [Section 4.4.1, RFC 6121] UC: <presence> <show>away</show> </presence>	2.13.3 IM-001640	R		T IO-4, IO-3
8-11	Upon receiving a presence stanza expressing updated availability, the user's server SHALL broadcast the full XML of that presence stanza to the contacts who meet all of the following criteria [Section 4.4.2, RFC 6121]: a. The contact is in the user's roster with a subscription type of "from" or "both." b. The last presence stanza received from the contact during the user's presence session was NOT of type "unsubscribe."	2.13.3.1 IM-001650	R		T IO-4, IO-3
8-12	The user's server SHALL also send the presence stanza to all of the user's available resources (including the resource that generated the presence notification in the first place). [Section 4.4.2, RFC 6121]	2.13.3.1 IM-001660	R		T IO-4, IO-3
8-13	Upon receiving presence from the user, the contact's server SHALL deliver the user's presence stanza to all of the contact's available resources. [Section 4.4.3, RFC 6121]	2.13.3.2 IM-001670	R		T IO-4, IO-3
8-14	From the perspective of the contact's client, there is no significant difference between initial presence broadcast and subsequent presence broadcast, so the contact's client SHALL follow the rules for processing of inbound presence defined under Section 2.13.1.4, Client Processing of Inbound Initial Presence. [Section 4.4.4, RFC 6121]	2.13.3.3 IM-001680	R		T IO-4, IO-3
8-15	Before ending its presence session with a server, the user's client SHALL gracefully become unavailable by sending unavailable presence, i.e., a presence stanza that possesses no „to" attribute and that possesses a „type" attribute whose value is "unavailable." The unavailable presence stanza SHALL NOT contain the <priority/> element or the <show/> element, since these elements apply only to available resources. [Section 4.5.1, RFC 6121] UC: <presence type='unavailable'/>	2.13.4.1 IM-001690	R		T IO-4, IO-3

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
8-16	The user's server SHALL NOT depend on receiving unavailable presence from an available resource, since the resource can become unavailable ungracefully (e.g., the resource can be timed out by the server because of inactivity). [Section 4.5.2, RFC 6121]	2.13.4.2 IM-001700	R		T IO-4, IO-3
8-17	If an available resource becomes unavailable for any reason (either gracefully or ungracefully), the user's server SHALL broadcast unavailable presence to all contacts that meet all of the following criteria [Section 4.5.2, RFC 6121]: a. The contact is in the user's roster with a subscription type of "from" or "both." b. The last presence stanza received from the contact during the user's presence session was not of type "error" or "unsubscribe."	2.13.4.2 IM-001710	R		T IO-4, IO-3
8-18	If the unavailable notification was gracefully received from the client, then the server SHALL broadcast the full XML of the presence stanza. [Section 4.5.2, RFC 6121]	2.13.4.2 IM-001720	R		T IO-4, IO-3
8-19	The user's server SHALL also send the unavailable notification to all of the user's available resources (including the resource that generated the presence notification in the first place). [Section 4.5.2, RFC 6121]	2.13.4.2 IM-001730	R		T IO-4, IO-3
8-20	If the server detects that the user has gone offline ungracefully, then the server SHALL generate the unavailable presence broadcast on the user's behalf. [Section 4.5.2, RFC 6121]	2.13.4.2 IM-001740	R		T IO-4, IO-3
8-21	Upon receiving an unavailable notification from the user, the contact's server SHALL deliver the user's presence stanza to all of the contact's available resources. [Section 4.5.3, RFC 6121]	2.13.4.3 IM-001750	R		T IO-4, IO-3
8-22	From the perspective of the contact's client, there is no significant difference between initial presence broadcast and unavailable presence broadcast, so the contact's client SHALL follow the rules for processing of inbound presence defined under Section 2.13.1.4, Client Processing of Inbound Initial Presence. [Section 4.5.4, RFC 6121]	2.13.4.4 IM-001760	R		T IO-4, IO-3
8-23	To specify a particular availability sub-state, a client implementation SHALL support the <show/> element within a presence stanza. A presence stanza SHALL NOT contain more than one <show/> element. The XML character data of the <show/> element is not human-readable. The XML character data SHALL be one of the following [Section 4.7.2.1, RFC 6121]: · away – The entity or resource is temporarily away. · Chat – The entity or resource is actively interested in chatting. · Dnd – The entity or resource is busy (dnd = "Do Not Disturb"). · Xa – The entity or resource is away for an extended period (xa = "extended Away"). NOTE: If no <show/> element is provided, the entity is assumed to be online and available. [Section 4.7.2.1, RFC 6121]	2.13.4.4 IM-001770	R		T IO-4, IO-3
8-24	When a user's client is engaged in a chat session with a contact, the user's client SHALL send a message of type "chat" and the contact's client SHALL preserve that message type in subsequent replies. [Section 5.1, RFC 6121]	2.13.5.1 IM-001780	R		T IO-4, IO-3
9	**2.14 – Exchanging Messaging**				
9-1	The user's client SHALL be capable of including a <thread/> element with its initial message, which the contact's client SHALL also preserve during the life of the chat session. The primary use of the XMPP <thread/> element is to uniquely identify a conversation thread or "chat session" between two entities instantiated by <message/> stanzas of type „chat". [Section 5.1, RFC 6121]	2.14.1 IM-001790	R		T IO-5
9-2	The user's client SHALL address the initial message in a chat session to the bare JID of the contact (i.e., <contact@domain>). Until and unless the user's client receives a reply from the contact, it SHALL continue sending any further messages to the contact's bare JID. Once the user's client receives a reply from the contact's full JID, it SHALL address its subsequent messages to the contact's full JID as provided in the „from" address of the contact's replies. [Section 5.1, RFC 6121]	2.14.1 IM-001800	R		T IO-5
9-3	The contact's client SHALL address its subsequent replies to the user's full JID <user@domain/resource> as provided in the „from" address of the initial message. [Section 5.1, RFC 6121]	2.14.1 IM-001810	R		T IO-5
9-4	An instant messaging client SHALL specify the intended recipient for a message stanza by providing the JID of the intended recipient in the „to" attribute of the <message/> stanza. [Section 5.2.1, RFC 6121]	2.14.1 IM-001820	R		T IO-5
9-5	An instant messaging client SHALL specify the intended recipient for a message stanza by providing the JID of the intended recipient in the 'to' attribute of the <message/> stanza (Section 5.2.1, RFC 6121).	2.14.2.1	R		

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
9-6	An instant messaging client SHALL support all of the following message types [Section 5.2.2, RFC 6121]: a. chat. The value "chat" indicates that the message is sent in the context of a one-to-one chat session. Typically, a receiving client will present/display messages of type "chat" in an interface that enables one-to-one chat between the two parties, including an appropriate conversation history. b. error. The value "error" indicates that the message is generated by an entity that experienced an error in processing a message received from another entity. NOTE: A client that receives a message of type "error" SHOULD present an appropriate interface informing the sender of the nature of the error. c. group chat. The value "group chat" indicates that the message is sent in the context of a multiuser chat environment. Typically, a receiving client will present a message of type "group chat" in an interface that enables many-to-many chat between the parties. d. normal. The value "normal" indicates that the message is a standalone message that is sent outside the context of a one-to-one conversation or group chat, and to which it is expected that the recipient will reply. Typically, a receiving client will present a message of type "normal" in an interface that enables the recipient to reply, but without a conversation history. The default value of the „type‟ attribute is "normal." NOTE: Support for the following message type is defined as recommended. e. headline. The value "headline" indicates that the message provides an alert, a notification, or other information to which no reply is expected (e.g., news headlines, sports updates, near-real-time market data, and syndicated content). Because no reply to the message is expected, typically a receiving client will present a message of type "headline" in an interface that appropriately differentiates the message from standalone messages, chat messages, or group chat messages (e.g., by not providing the recipient with the ability to reply).	2.14.2.2 IM-001830	R		T IO-5
9-7	If an application receives a message with no „type‟ attribute or the application does not understand the value of the „type‟ attribute provided, it SHALL consider the message to be of type "normal" (i.e., "normal" is the default). [Section 5.2.2, RFC 6121]	2.14.2.2 IM-001840	R		T IO-5
9-8	A client SHALL be capable of populating a stanza with the element. The element contains human-readable XML character data that specifies the textual content of the message.	2.14.2.3 IM-001850	R		T IO-5
10	**2.15 - Conformance Requirements in RFC 6120 and RFC 6121**				
10-1	Section 15 of RFC 6120 and Section 13 of RFC 6121 describe a protocol feature set that summarizes the conformance requirements associated with these two specifications. In the event of a discrepancy between Section 15 of RFC 6121 or Section 13 of RFC 6121 and the UC XMPP 2013 Specification, the explicit requirements defined in the UC XMPP 2013 Specification take precedence.	2.15	R	R	L
11	**2.16 - XMPP Extensions**				
11-1	The protocol specifications referenced within Table 2.16-1, DoD XMPP Protocol Suite, constitute a mandatory protocol suite (i.e., for the purpose of compliance testing and certification; support for these extensions is defined as REQUIRED). Regarding the specifications defined in Table 2.16-1, DoD XMPP Protocol Suite, client and server implementations SHALL comply with all requirements defined as “MUST”, “SHALL”, “REQUIRED”, “MUST NOT”, “SHALL NOT.” It is also expected that vendors will likewise implement requirements defined as “SHOULD” or “SHOULD NOT” except where there may exist valid reasons in particular circumstances to ignore a particular requirement.	2.16	R	R	L
11-2	To better enable multivendor interoperability, to facilitate full feature functionality, and to address specific security requirements, some of the requirements defined as “SHOULD”, “RECOMMENDED”, “SHOULD NOT”, “NOT RECOMMENDED”, “MAY”, or “OPTIONAL” in the above XMPP extensions have been redefined in this specification to reflect requirement levels associated with the following terminology: “MUST”, “SHALL”, “REQUIRED”, “MUST NOT”, or “SHALL NOT.” These elevated requirements are explicitly defined in Table 2.16-2. Also, where there may be some degree of ambiguity in a commercial standard regarding whether or not support for a particular capability or feature is REQUIRED, Table 2.16-2, Elevated/Clarified Requirements, adds explicit clarification.	2.16.1	R	R	L
12	**2.17 - XML Usage**				
12-1	XMPP client and server implementations SHALL comply with the mandatory requirements defined in Section 11 of RFC 6120.	2.17 IM-001860	R		L
13	**2.18 - DIFFSERV Code Point (DSCP) Requirements**				

ID	REQUIREMENT	XMPP 2013 / UCR Ref	XMPP Server Client	XMPP Gateway	LoC/ TP ID
13-1	XMPP client and server implementations shall class mark XMPP traffic consistent with the code point value defined for ROUTINE Low-Latency Data as per the DSCP Assignments defined in Section 6 of UCR 2013.	2.18 IM-001870	R		T IO-1
14	**UCR 2013, Section 5 – Internet Protocol version 6 Requirements**				
14-1	Must be IPv6-capable. Use guidance in Table 5.2-4 for NA/SS.	Table 5-2.1	R	R	L

LEGEND:

.mil	.mil Domain	L	Letter of Compliance
A	Address	LoC	Letter of Compliance
AAAA	Authentication, Authorization, Accounting and Auditing	O	Optional
C	Conditional	R	Required
CR	Certification Requirement	Ref	Reference
DISN	Defense Information Systems Network	RFC	Request For Comments
DNS	Domain Name Server	SRV	Server Location Service
DSCP	DIFFSERV Code Point	SUT	System under Test
en	English Language	T	Testable
FR	Functional Requirement	TCP	Transmission Control Protocol
HTTP	Hyper Text Transfer Protocol	TLS	Transport Layer Security
IA	Information Assurance	TP	Test Procedure
id	identification	UC	Unified Capabilities
ID	Identification	v	version
IM	Instant Message	XEP	XMPP Extension Protocols
IP	Internet Protocol	XML	eXtensible Markup Language
JID	Jabber Identifier	XMPP	Extensible Messaging and Presence Protocol

Table 3-6. XMPP Extensions Elevated/Clarified Requirements

REQUIREMENT	Reference Document Section	XMPP Server Client	XMPP Gateway	LoC/ TP ID
REFERENCE DOCUMENT: XEP-0045 Multi-User Chat				
Implementations SHALL provide support for the 'Visitor' role.	5.1	R		T IO-7
Implementations SHALL provide support for the 'Admin', 'Member', and 'Outcast' affiliation.	5.2	R		T IO-7
Implementations SHALL support the following capabilities (as defined in Sections 6.1, 6.2, and 6.3): 1. Discovering Component Support for MUC 2. Discovering Rooms 3. Querying for Room Information	6.1 6.2 6.3	R		T IO-7
Implementations SHALL support the following room types: 1. Both Persistent or Temporary 2. Non-Anonymous 3. Password-Protected and Unsecured 4. Both Members-Only and Open 5. Moderated and Un-moderated	3 4.2 7.1.5 7.1.6 7.1.7 7.1.8	R		T IO-7
Implementations SHALL support the sending of Discussion History to a new occupant (as defined in Sections 7.1.15). NOTE: "Whether such history is sent, and how many messages comprise the history, shall be determined by the chat service implementation or specific deployment."	7.1.15	R		T IO-7
Implementations SHALL support a user's ability to: 1. Enter a Room 2. Exit a Room 3. Change Availability Status 4. Send a Private Message 5. Send a Message to All Occupants 6. Register with a Room 7. Request Voice	7.1 7.2 7.4 7.5 7.6 7.8 7.9 7.10 7.13	R		T IO-7
Implementations SHALL support the ability of a Moderator to perform the following privileges: 1. Modify the subject 2. Kick a participant or visitor from the room 3. Grant or revoke voice in a moderated room 4. Modify the list of occupants who have voice in a moderated room	8.1 through 8.6	R		T IO-7
Implementations SHALL support the ability of an Admin to perform the following privileges: 1. Ban a user from the room 2. Modify the list of users who are banned from the room 3. Grant or revoke membership 4. Modify the member list 5. Grant or revoke moderator privileges 6. Modify the list of moderators 7. Approve Registration Requests	9.1 through 9.9	R		T IO-7
Implementations SHALL support the ability of an Owner to create a room and to change defining room configuration settings (as defined in Section 10.1 and 10.2)	10.1 and 10.2	R		T IO-7
Implementations SHALL support the ability of an Owner to perform the following privileges (as defined in Section 10): 1. Grant or revoke ownership privileges 2. Modify the owner list 3. Grant or revoke administrative privileges 4. Modify the Admin list 5. Destroy a room	10.3 through 10.9	R		T IO-7
REFERENCE DOCUMENT: XEP-0030 Service Discovery				
Implementation SHALL provide support for: 1. Discovering information about an entity as defined in Section 3 [XEP-030] 2. Discovering the items associated with an entity as defined in Section 4 [XEP-030]	4 and 3	R	R	T IO-7
NOTE: Table 2.16-2, Elevated/Clarified Requirements, ONLY addresses functionality where the associated requirement level has been elevated (e.g., from a "SHOULD" to a "SHALL") or where there was a need to explicitly clarify whether support for a particular capability or feature is REQUIRED.				